УДК 004.422 **doi: 10.26906/SUNZ.2022.1.081**

І. К. Сисоєв, В. В. Гавриленко

Національний транспортний університет, Київ, Україна

АДАПТИВНИЙ АЛГОРИТМ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В ДОДАТКАХ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ КОНТЕЙНЕРИЗАЦІЇ

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, при досягненні яких такий алгоритм можна буде використовувати замість вже існуючих.

Ключові слова: синхронізатор, балансувальник навантаження, адаптивний алгоритм, контейнеризація, обчислювальний вузол.

Вступ

Балансування навантаження (Load Balancing) застосовується для оптимізації виконання паралельних обчислень за допомогою паралельної обчислювальної системи, яка передбачає рівномірне навантаження обчислювальних вузлів.

При появі нових завдань програмне забезпечення, що реалізує балансування, має прийняти рішення про те, на якому обчислювальному вузлі слід виконувати обчислення, пов'язані з цією новою задачею. Балансування навантаження також передбачає перенесення частини обчислень з найбільш завантажених обчислювальних вузлів на менш завантажені [1] .

Постановка задачі. В сфері створення високо навантажених додатків, що розгорнуті в хмарі, гостро стоїть питання максимально ефективного використання ресурсів, за які користувач постійно сплачує. Неефективність алгоритмів балансування призводить до часткового простою ресурсів. Реалізація більш досконалого алгоритму надасть можливість користувачам заощаджувати кошти на невикористаних ресурсах.

Реалізація такої паралельної обчислювальної системи вимагає розробки алгоритмів синхронізації об'єктів, які функціонують в різних вузлах обчислювальної системи. І навпаки, ефективність реалізації алгоритмів синхронізації залежить від збалансованості навантаження по вузлах обчислювальної системи.

Дисбаланс навантаження може виникнути з кількох причин:

- неоднорідність структури паралельного застосування, тобто різні запити процесу вимагають різних обчислювальних потужностей;
- неоднорідність структури обчислювального комплексу, тобто різні обчислювальні вузли володіють різною продуктивністю;
- неоднорідність структури взаємодії між вузлами обчислювальної мережі, тобто лінії зв'язку між вузлами можуть мати різні характеристики пропускної спроможності [2].

На сьогоднішній день універсального методу боротьби з дисбалансом навантаження не існує.

Аналіз літератури. Методи балансування поділяють на статичні та динамічні.

Статичне балансування виконується до початку виконання паралельного додатка. При розподілі логічних процесів по процесорам використовується досвід попередніх виконань (так зване навчання на історичних даних). Однак розміщення логічних процесів по вузлах обчислювальної мережі попередньо може не дати позитивних результатів.

Це пояснюється тим, що:

- обчислювальне середовище, в якому відбувається виконання програми, може змінитися, тобто один або кілька обчислювальних вузлів можуть вийти з ладу;
- обчислювальний вузол, на якому виконується паралельне додаток, може бути зайнятий іншими обчисленнями;
- навантаження на обчислювальні вузли є непередбаченим (аномальна активність користувачів, DDoS атаки).

Динамічне балансування передбачає перерозподіл навантаження на обчислювальні вузли під час виконання програми. Для такого балансування використовують програмне забезпечення, яке визначає:

- завантаження обчислювальних вузлів;
- пропускну спроможність ліній зв'язку між вузлами;
- частоту вхідних запитів;
- ресурсоємність вхідних запитів;
- кількість і завантаженість обчислювальних вузлів.

Динамічні методи зазвичай застосовують, якщо час, необхідний на балансування, набагато менше часу виконання завдання [3]. Динамічна задача балансування зазвичай включає в себе не тільки розподіл навантаження по обчислювальним вузлам, а й,

з огляду на особливості алгоритму балансування, вибір оптимального числа обчислювальних вузлів. Балансування навантаження може виконуватися програмно або апаратно, централізовано або децентралізовано.

Алгоритм динамічного балансування визначає на який обчислювальний вузол адресувати запит під час роботи системи.

Такий підхід дозволяє реагувати на зміну стану додатку. Однак динамічне балансування тягне за собою додаткові витрати часу на збір статистичних даних про стан додатку, на аналіз даних і на прийняття рішень.

Невирішені питання. Ключовим завданням при створенні адаптивного алгоритму є реалізація механізму балансування, затрати на прийняття рішення якого були б значно меншими за типові запити на виконання. Від ефективності цього механізму буде залежати доцільність використання адаптивного алгоритму в перевагу існуючим.

Мета статті. Метою даної статті є формування науково-обґрунтованого підходу до створення адаптивного алгоритму балансування навантаження в системах з використанням технології контейнеризації, для подальших реалізації реальних алгоритмів на основі теоретичних обґрунтувань.

Виклад основного матеріалу

Контейнеризований додаток являє собою сукупність однакових контейнерів з екземпляром додатку. Екземпляри розподіляються за різними обчислювальним вузлам і функціонують паралельно. Розподіляються запити між обчислювальними вузлами таким чином, щоб завантаження обчислювальних вузлів була рівномірним.

Однак при виконанні паралельного додатка може виникнути конфлікт між збалансованим розподілом об'єктів по обчислювальних вузлах і низькою швидкістю обмінів даними між цими об'єктами. Деякі обчислювальні вузли можуть простоювати, в той час як інші будуть перевантажені, якщо комунікація між обчислювальними вузлами ведеться на низькій швидкості [4].

З іншого боку, витрати на комунікацію можуть бути великі для збалансованої системи. Саме тому метод балансування повинен бути підібраний таким чином, щоб обчислювальні вузли були завантажені рівномірно, а швидкість обміну даними між ними була оптимальною.

Балансування навантаження - це механізм, який використовується для розподілення завдання між обчислювальними вузлами. Балансування і перенесення навантаження використовують для підвищення продуктивності паралельної системи [5]. Через неоднорідність обчислювального середовища, один алгоритм може добре працювати в одній паралельній системі і погано в інший.

В сучасних тенденціях розробки розподілених додатків спостерігається все більша практика використання хмарних середовищ для розгортання таких як (Azure, Google Cloud, Azure тощо) з застосуваннями технології контейнеризації.

При такому підході такі причин дисбалансу навантаження як неоднорідність структурі обчислювального комплексу та неоднорідність структурі взаємодії між вузлами нівелюються. За допомогою використання технології контейнеризації яка дозволяє розгортати обчислювальні вузли як незалежні один від одного контейнери які мають однакову, попередньо сконфігуровану обчислювальну потужність, досягається однорідність середовища для обчислення запитів. Всі контейнери підпорядковані одному вузлу синхронізатору (Load balancer - балансувальник навантаження) синхронізатор реалізує алгоритм розподілення навантаження на підпорядковані вузли. При використанні технології контейнеризації окрім розподілення навантаження синхронізатор також може додавати нові обчислювальні вузли або вимикати не задіяні для економії коштів. Кожен обчислювальний вузол розгорнутий в хмарі має певну вартість згідно моделі тарифікації провайдера хмарного середовища. Саме економія коштів стає додатковим стимулом для максимально ефективного використання наявних ресурсів і оптимальне додавання нових.

Тому для ефективного балансування навантаження необхідно щоб алгоритм максимально задовольняв двом критеріям:

1) максимально ефективне використання наявних обчислювальних вузлів;

2) оптимальний алгоритм додавання або вимикання обчислювальних вузлів.

Під ефективним використанням вузлів розуміється те, що ресурси обчислювального вузла мають бути використанні на 90-95 відсотків, тобто на нього повинно бути відправлено стільки операцій, скільки він зможе ефективно виконати без втрат в часі виконання операції.

Під оптимальним алгоритмом додавання вузлів розуміється – що синхронізатор повинен додавати новий вузол тільки тоді коли вже неможливо буде виконати операціях на наявних вузлах без втрати в часі виконання операції.

Для досягнення заданих критеріїв пропонується розробка адаптивного алгоритму балансування який приймав би рішення на основі комплексного показнику ресурсоємності обчислювального вузла, тобто величини яка б характеризувала що 1 вузол може обробити N операції певного типу без втрати в часі виконання операції. Для цього нам потрібно визначити ресурсоємність нашого вузла в RPS (Requests Per Second - кількість запитів за одну секунду). Також визначити RPS для кожного типу запиту. Тип запиту визначається по його URI який є унікальним ідентифікатором запиту і гарантує що при правильній архітектурі додатку кожен запит по тому й самому URI буде використовувати приблизно однакову кількість ресурсів сервера. Точність визначення RPS для кожного запиту буде на пряму впливати на ефективність синхронізатора при додаванні або вимиканні обчислювальних вузлів.

Алгоритм розподілення запиту з одного обчислювального вузла на інший досить складний. Однак введення деяких припущень може знизити складність цього алгоритму. нехай:

- розподілена система об'єднує кілька різних типів даних (є гетерогенною);
- користувач може інтерактивно взаємодіяти з машиною в будь-який момент часу;
- кількість обчислювальних вузлів в мережі контролюється тільки синхронізатором без втручання користувача в повністю автоматичному режимі;
- можлива зміна обчислювальної мережі;
- затримки на обмін даними між вузлами залишаються незмінними.

Для реалізації алгоритму пропонується реалізувати трирівневу систему прийняття рішень і збору даних.

На першому рівні пропонується наділити синхронізатор можливістю збирати і обробляти статистичні данні по навантаженням від самих вузлів так активувати паралельний процес прогнозування вхідного навантаження в залежності від вхідного навантаження.

Також синхронізатор повинен провести оцінку вхідного запиту для визначення його ємності на основі статистичних даних для прийняття рішення про адресацію цього запиту на існуючий обчислювальний вузол, якщо ємність вузла достатня для виконання запиту, або адресувати запит на нещодавно створений ввіходячи з донних отриманих від системи прогнозування обчислювальний вузол.

На другому рівні пропонується створити систему прогнозування навантаження яку б задіяв синхронізатор відправляючи їй паралельно вхідні запити та данні від обчислювальних вузлів про затрачену ємність на виконання ємність.

В життєвому циклі додатку навантаження в часі в основному не однорідне а його піки часто співпадають с піками активності користувачів або заздалегідь запланованими піковими активностями, система прогнозування цих навантажень дасть нам можливість не створювати додаткові обчислювальні вузли в реальному часі, а додавати їх за декілька хвилин до того як додатковий обчислювальний вузол нам знадобиться.

На третьому рівні пропонується при створенні нового контейнеру розгортати додатковий агент разом з обчислювальним вузлом який в фоновому режимі збирав би данні по вхідних запитах і ресурсах витрачених на його обробку.

Відправка цих донних на синхронізатор, необхідна для точного визначення ємності запиту, збір цих донних пропонується перенести на сторону обчислювальних вузлів для зменшення навантаження на синхронізатор щоб максимально скоротити час прийняття рішення, так як збір статистики на фоновому рівні не повинно забирати багато ресурсів у обчислювального вузла.

Висновки

В результаті реалізації такого алгоритму очікуються досягнути переваг, які дозволять використовувати таку систему замість існуючих підходів і алгоритмів.

По-перше, ресурси, що витрачені на прийняття рішень, не повинні бути більшими за ресурси, які необхідні на виконання цього запиту.

По-друге, доступність додатку має бути на рівні 99.99% без втрати швидкодії при додаванні або вимиканні нових обчислювальних вузлів.

По-третє, наявність незадіяних вузлів та вузлів з частковою завантаженістю повинна бути мінімізована.

СПИСОК ЛІТЕРАТУРИ

1. Andrey Vladimirov, Ryo Asai, Vadim Karpusenko. Parallel Programming and Optimization with Intel Xeon Phi Coprocessors, 2015.
2. Load Balancing in Parallel Computers. Електронний ресурс. Режим доступу: http://www.inspirenignite.com/load-balancing-in-parallel-computers/
3. Maurice Herlihy , Nir Shavit, Victor Luchangco , Michael Spear. The Art of Multiprocessor Programming, 2020.
4. Christoforos N Hadjicostis, Alejandro D Dominguez-Garcia, Themistokis Charalambous. Distributed Averaging and Balancing in Network Systems, 2018.
5. Балансування навантаження у розподілених системах. Електронний ресурс. Режим доступу http://intuit4.intuit.ru/studies/courses/1146/238/lecture/3287?page=1

Надійшла (received) 23.12.2021
Прийнято до друку (accepted for publication) 16.02.2022

Adaptive load balancing algorithm for applications using containerization technology

I. Sysoiev, V. Gavrilenko

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.

Keywords: synchronizer, load balancer, adaptive algorithm, containerization, compute node.