1) Еволюція, яка вивчається в даній теорії, необов'язково повинна бути біологічною еволюцією. Вона також може бути і культурною, тобто відображати зміни в нормах і переконаннях в часі.

2) Раціональні припущення, що лежать в основі еволюційної теорії ігор, у багатьох випадках більш підходять для моделювання соціальних систем, ніж припущення, що лежать в традиційній теорії ігор.

3) Еволюційна теорія ігор є динамічною теорією ігор, що знову ж таки вигідно відрізняє її від традиційної теорії ігор.

Однією і найпростіших моделей еволюційної теорії ігор є гра «El Farol bar problem», що часто застосовується при моделюванні економічних процесів і відноситься до підкласу ігор «Minority games». Найпростіший випадок такої гри складається з N непарного числа гравців, які вибирають один з двох варіантів можливих рішень протягом кожного раунду гри. Таким рішенням може бути, наприклад, покупка або продаж акцій, або інших активів. Гравець вважається виграв, якщо виявляється в меншості. І таким шляхом в результаті такої гри формується меншість гравців, яка завдяки своїй стратегії, вдаліше інших передбачає результат гри. Така гра також отримала назву задача бару «El Farol bar problem» і була вперше описана Вільямом Артуром в 1994 році. Різні моделі, засновані на іграх меншини, активно використовуються для аналізу фінансових ринків. Крім цього такі ігри дозволяють моделювати макроекономічні процеси.

Висновки. В ході виконання роботи було проведено дослідження динаміки еволюційних процесів на прикладі економічної стратегічної гри «El Farol bar problem».

ПЕРЕЛІК ПОСИЛАНЬ:

1. Alexander, J. McKenzie, Evolutionary Game Theory // The Stanford Encyclopedia of Philosophy. – 2009. [Электронный ресурс]. URL: http://plato.stanford.edu/entries/game-evolutionary/

2. Диксит А. Стратегические игры / А. Диксит, С. Скит, Д. Рейли., 2017. – 880 с.

УДК 004.9

ПРОЕКТУВАННЯ ТА ОПТИМІЗАЦІЯ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ ВХІДНОГО ТРАФІКУ СЕРВЕРА

Д.Е. Чернорот, І.С. Дмитрієва
(Україна, Дніпро, Національна металургійна академія України)

Постановка проблеми: У наш час, коли інтернет займає більшу частину сфер життя, неможливо уявити собі будь-яке діло (компанію, бізнес), яке не буде використовувати для виконання частини роботи інтернет. Зберігання даних, віддалена робота або, що зараз найбільше використовується у більшості різноманітної промисловості, сайти використовують певні ресурси, а саме

сервери. За рахунок нашого темпу життя ми не маємо часу на те, щоб очікувати, доки ми отримаємо відповідь на наш запит на файл, чи доки відкриється веб-сторінка. Саме тому з технічного боку постає питання про те, як саме забезпечити швидкий відклик від сервера, при цьому зекономивши кошти на масштабуванні обчислювальних ресурсів?

Аналіз досліджень: Для найбільш вдалого розділення запитів до сервера найкраще використовувати балансування трафіку. Буває декілька видів балансування вхідного трафіку на сервер. Але спочатку необхідно розуміти, що балансування має задовольняти певні умови[1]:

– балансування повинно задовольняти вимогам справедливості - будь-який запит, що прийшов на сервер у нашому кластері, повинен бути обслужений, а не кинутий без відповіді;

– балансування повинно задовольняти вимогам передбачуваності - ми повинні заздалегідь чітко розуміти, який з алгоритмів балансування в певній ситуації ми зможемо використовувати;

– балансування повинно бути здатним до масштабування - при виявленні різкого збільшення трафіку (а, відповідно, і навантаження) система повинна забезпечувати стабільну роботу нашого сервісу;

– скорочення часу відповіді - при використанні балансування ми повинні в підсумку отримати скорочення часу виконання запиту до сервера, тобто, як тільки запит приходить в нашу систему, ми повинні якомога швидше його обслужити, дати на нього відповідь;

– ефективність - ми повинні забезпечити таку роботу балансування, при якій всі сервера, що знаходяться в нашому кластері, працювали б приблизно однаково, навантаження було рівномірним.

За географічною ознакою балансування умовно можемо розділити на два види:

– локальна - якщо наші сервери розміщені всередині одного дата-центру;

– глобальна - наші сервера розкидані по різних дата-центрам.

Локальне балансування можна застосовувати на канальному рівні (з використанням окремого балансувальника і без нього), на мережевому рівні або на транспортному рівні. Дані способи найбільш поширені при локальної балансуванню.

Балансування на канальному рівні виконується за рахунок наступного: ми беремо і навішуємо на якийсь спеціалізований інтерфейс всіх наших серверів однуй ту ж IP-адресу нашого ресурсу, на який будуть приходити запити, і з якого будуть йти відповіді. Але на ARP-запит з цієї IP-адреси сервера не повинні відповідати. І ми навішуємо таку ж IP-адресу на наш балансувальник, відповідно, на нього будуть приходити запити, і відправлятися відповіді з нього, і він же буде відповідати на ARP запити. Таким чином, отримуючи запит від клієнта, наш балансувальник вибирає за певним алгоритмом той чи інший сервер, який буде обробляти цей запит, підміняє destination MAC і відправляє його на обробку на даний сервер. Сервер його у себе обробляє, і, так як ми не

робили підміну заголовків на мережевому рівні, то безпосередньо, минаючи балансувальник, сервер відразу відповідає клієнту через наш шлюз [2].

Реалізувати подібне балансування без окремого балансувальника (і в підсумку скоротити витрати) можна наступним чином: нам необхідно перетворити вхідний unicast запит в broadcast, або в multicast. Робиться це в такий спосіб: всі сервера повинні на ARP запит відповідати однією і тою самою MAC-адресою, тобто, це може бути або неіснуючий MAC-адресу, або якийсь мультикастового. Або ми можемо навісити цей мультикастового MAC-адресу на наш шлюз. Відповідно, запит приходить на наш ресурс, і шлюз його просто розмножує до всіх серверів, таким чином запити надходять на всі сервери одночасно, і кожен сервер повинен сам розуміти, чи повинен він відповідати на запит чи ні.

Балансування на мережевому рівні має досить схожий механізм з балансуванням на канальному рівні. Єдина відмінність в тому, що в даному випадку при отриманні вхідного запиту наш балансувальник підміняє destination IP, переправляючи його на той сервер, який буде обробляти запит. Сервер отримує його, обробляє і повинен передати його назад балансувальник, щоб той виконав зворотну заміну[3].

Балансування на транспортному рівні. Тут дуже тонка грань, яка відрізняє балансування на мережевому від балансування на транспортному рівні. Для простоти скажемо, що в даному виді балансування використовуються при балансуванні навантаження вхідні порти джерела і адресата.

В балансуванні DNS часто застосовують так званий алгоритм Round Robin. Це найпростіший механізм балансування, за допомогою якого можна балансувати будь-які системи, в яких доступ до сервісу відбувається по імені. Саме цей метод у зв’язці з іншими видами балансування буде розглянуто в рамках дипломної роботи, виходячи з наявних ресурсів. Суть даного методу в наступному: на DNS сервер додається кілька A-записів з різними IP-адресами всіх наших серверів, і сервер сам буде в циклічному порядку видавати ці адреси. Тобто, перший запит отримає перший сервер, другий запит - другий сервер, третій запит - третій сервер і так далі.

Наступний алгоритм – це алгоритм проксіювання. Суть даного алгоритму полягає в тому, що в якості балансувальника застосовується так званий «розумний» проксі. Тобто якщо балансувальник отримує запит до нашого ресурсу, він аналізує заголовки прикладного рівня і, відповідно, він може розуміти до якого ресурсу прийшов запит на наш балансувальник, і направити запит на той чи інший сервер, на якому цей ресурс міститься. Також при отриманні даного запиту балансувальник може додавати в заголовки HTTP, наприклад, інформацію про те, з якого IP прийшов клієнт. Корисно це для того, щоб сервер знав, куди його потім згодом відправляти, і з ким він працює. Виконавши запит сервер передає його назад на балансувальник, балансувальник виконує необхідні маніпуляції з новими заголовками або третього рівня, або сьомого рівня і віддає його клієнту [1].

Redirect запитів. Redirect запитів має досить обмежене застосування – застосовується, в основному, для глобального балансування, і, зокрема, для

HTTP він добре застосується. Суть його полягає в тому, що ми отримуємо запит від клієнта на наш балансувальник, балансувальник відповідає йому редіректом на наш сервер, на якому містяться ресурси. Наприклад, отримуючи запит по HTTP, балансувальник відповідає йому у відповідь кодом 302 move temporary із зазначенням адреси того сервера, на який далі буде ходити наш клієнт.

Балансування на базі Anycast. Цей алгоритм балансування не вимагає ніякого налаштування з боку клієнта, і суть його полягає в наступному: ми з різних географічних ділянок анонсуємо один і той же префікс мережі. Таким чином, кожен запит клієнта маршрутизується на найближчий до нього сервер, який буде його обробляти.

Цілі: Необхідно, використовуючи методи балансування, розробити систему, що буде з найменшими вкладеннями балансувати трафік и витримувати велике навантаження з боку запитів клієнтів.

Висновки: З найменшими витратами можливо балансувати трафік за допомогою алгоритму RoundRobin, що використовується при балансуванні DNS. Цей алгоритм є найбільш вдалим з огляду на масштабування. За допомогою налаштування параметрів ми зможемо досягти оптимального відклику від серверів при отриманні запиту. При цьому для масштабування не буде необхідності правити повторно конфігурації на серверах, буде досить додати новий запис з IP адресою додаткового сервера и таким чином ми зможемо збільшити ресурси нашого кластеру у самий короткий термін без простою усього кластера. Таким чином ми досягнемо також зменшення ризику відмови нашого кластеру, тобто відмовостійкість наших обчислювальних потужностей збільшиться на певний відсоток.

ПЕРЕЛІК ПОСИЛАНЬ:

1. Y. Hu, R. Blake, D. Emerson. An optimal migration algorithm for dynamic load balancing. Concurrency: Practice and Experience. – V.10(6). – 1998. P. 467–483.

2. E.G. Ignatenko, V.I. Bessarab, V.V. Turupalov, The algorithm of adaptive load balancing in cluster systems, Modeling and information technologies, Kyiv: IPME G.E. Puhova NAN of Ukraine, № 58, 2010, pp. 142-150.

3. Hisao Kameda, Lie Li, Chonggun Kim, Yongbing Zhang. Optimal Load Balancing in Distributed Computer Systems. Springer, Verlag London Limited.- London. - 1997. - P. 238