ARTICLE

DOI 10.36074/grail-of-science.27.05.2022.056

КОМБІНОВАНИЙ МЕТОД МАСШТАБУВАННЯ БАЗ ДАНИХ

Тетяна Іванівна Коробейнікова
канд. техн. наук, доцент,
доцент кафедри безпеки інформаційних технологій
Національний університет «Львівська політехніка», Україна

Захарченко Сергій Михайлович
канд. техн. наук, доцент, доцент кафедри обчислювальної техніки
Вінницький національний технічний університет, Україна

Анотація. *Метою даної роботи є дослідження роботи комбінованого методу масштабування баз даних для прискорення роботи серверних додатків під час високих навантажень та автоматизації процесів масштабування серверних систем.*

Ключові слова: *контейнеризація, комбінований метод масштабування комбінованих баз даних, механізм балансування навантаження в базі даних, масштабування веб-ресурсів.*

Комбінований метод масштабування баз даних. Бази даних завжди заповнені (зберігаються дані користувача/системи) [1-2], тому для їх масштабування потрібна підтримка цілісності/узгодженості даних, а це завдання непросте. Реплікація та шардінг є основними способами масштабування баз даних.

Реплікація бази даних – це метод, при якому екземпляр бази даних точно копіюється, передається або інтегрується з іншим місцем. Реплікація бази даних дозволяє скопіювати файл бази даних з головної системи керування базами даних (СУБД) і точно розгортати його в підпорядковану базу даних [3-4]. Існує два типи реплікації бази даних: Master-Slave і Master-Master (Cluster).

Реплікація Master-Slave вибирається, якщо більшість запитів до програми бази даних – «читані». У цьому випадку екземпляр бази даних Master є основним, його стан копіюється на підпорядковані вузли (Slave), тому програма може «записувати» дані тільки на головний вузол і «читати» дані як з Master, так і з Slave. Цей тип реплікації підвищує відмовостійкість бази даних, оскільки в будь-який момент часу існує кілька незалежних екземплярів усієї бази даних. У разі відмови підпорядкованого вузла – його легко замінити без втрати системи. У разі відмови провідного вузла – один з підлеглих вузлів стає новим провідним, а старий провідний замінюється новим відомим (рис. 1).

All rights reserved | Creative Commons Attribution 4.0 International License

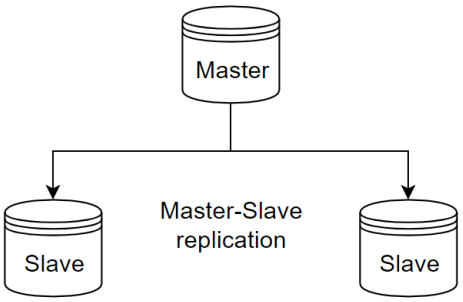

Рис. 1. Взаємодія екземплярів БД під час реплікації Master-Slave

Реплікація Master-Master дає можливість отримати доступ до будь-якого екземпляра як для «читання», так і для «запису», але цей тип реплікації підтримується невеликою кількістю СУБД (рис. 2).

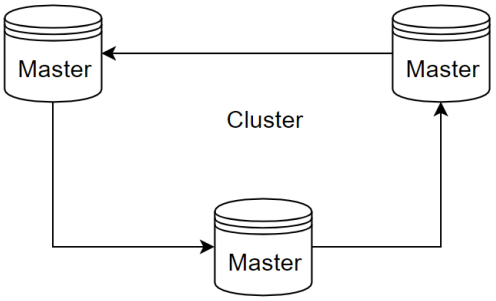

Рис. 2. Взаємодія екземплярів БД під час реплікації Master-Master

Шардинг бази даних – це метод, коли різні дані записуються в різні вузли бази даних, наприклад, кожна база даних може зберігати дані 1000 користувачів, і коли цей ліміт досягається, програма перемикається на новий екземпляр бази даних, тому ці сервери не є взаємопов'язані. Програма вибирає, з яким вузлом працювати. Шардінг є незалежним механізмом. Для підвищення відмовостійкості кожного шарда цей метод зазвичай поєднують з реплікацією Master-Slave, що в сумі методів дозволяє необмежено масштабувати комплекс баз даних (рис. 3).

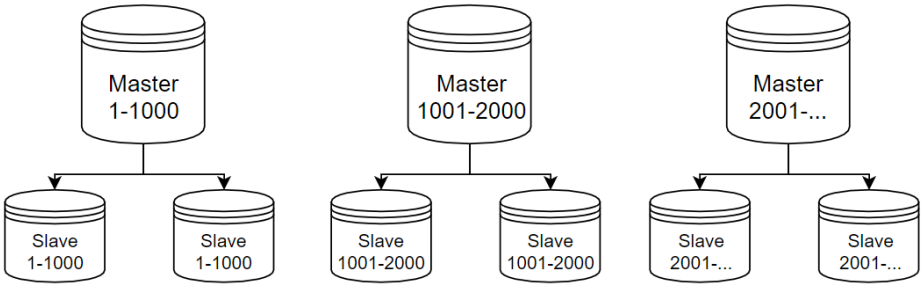

Рис. 3. Взаємодія екземплярів БД під час поєднання Sharding і Master-Slave реплікації

Залежно від середовища, де запущено додаток і база даних, процедура масштабування відрізняється, і автоматичне масштабування доступне лише в хмарних системах [5-6]. Існують такі середовища для запуску веб-ресурса:

– Фізичний сервер;
– Виділений сервер;
– Віртуальний сервер;
– Хмара;
– Service Mesh (мережа послуг).

Фізичний сервер знаходиться безпосередньо в організації, яка запускає веб-ресурс. Цей сервер обслуговується цією організацією. Фізичне обслуговування сервера включає моніторинг фізичних параметрів (ефективність охолодження, чистота, справність компонентів) і контроль системи (обслуговування ОС, оновлення програмного забезпечення, налаштування мережі та безпека мережі). Одна ОС зазвичай встановлюється на фізичному сервері і програми розгортаються в його середовищі, при високих навантаженнях програми конкурують за ресурси сервера.

Виділений сервер розташований у спеціалізованому дата-центрі (рис. 4). Фізичне обслуговування виділеного сервера забезпечує приймаюча компанія, а налаштування системи виконує адміністратор, який запускає програми на цьому сервері. Конфігурація віддалена, оскільки сервер може фізично знаходитися в будь-якому місці [7-8].

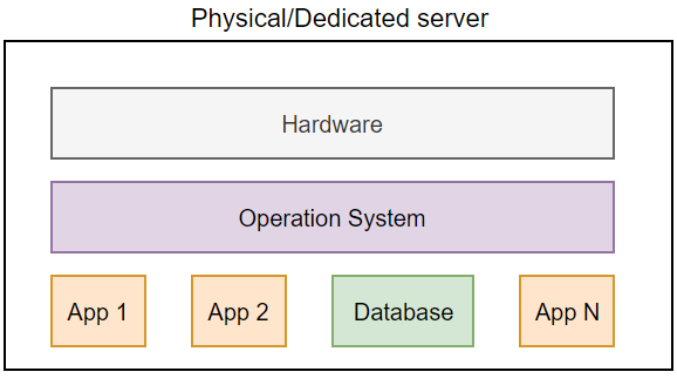

Рис. 4. Програми на фізичному / виділеному сервері (додатки конкурують за ресурси)

При використанні фізичного/виділеного сервера є дві стратегії масштабування додатків: заміна сервера на більш потужний або розгортання додаткового сервера та переміщення деяких програм на нього. Обидві операції вимагають часу та певного простою.

Хмара – це середовище, створене для полегшення запуску різних типів додатків, хмари бувають загальнодоступними та приватними. Публічні хмари використовуються всіма, хто платить за використаний час або зайняті ресурси. Приватні хмари створюються та обслуговуються підприємствами/компаніями. Додатки в хмарі зазвичай запускаються в контейнерах, де одна програма закрита в одному контейнері, що дозволяє запускати її незалежно вимогам до потужностей або середовища (рис. 5).

All rights reserved | Creative Commons Attribution 4.0 International License

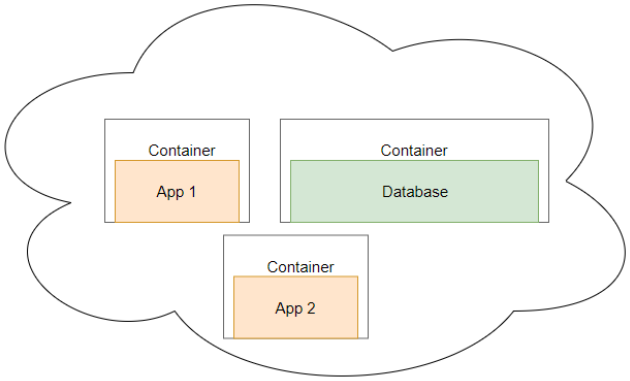

Рис. 5. Додатки в хмарі (програми не конкурують за ресурси)

Зазвичай в хмарах основним сервісом є послуга оренди віртуального сервера, додатковими послугами є розміщення баз даних, серверів черг, документоорієнтованих сховищ тощо [9]. Таким чином, ви можете запускати свою програму як контейнер на віртуальному сервері, не потребуючи налаштування. та контейнеризувати базу даних. Програми, що працюють у хмарі, масштабуються як по горизонталі, так і по вертикалі, змінюючи будь-які налаштування вручну [10]. У хмарі також є функція автоматичного масштабування, яку можна налаштувати в консолі керування хмарою, але спочатку потрібно визначити критерії, за якими буде відбуватися автомасштабування.

Service Mesh – це рівень абстракції, який дозволяє запускати програми та служби в контейнерах, не турбуючись про те, як програми підключаються до фізичного рівня. Ця система формує кластер з будь-якої кількості будь-яких серверів, фізично встановлених на них один раз, і дозволяє запускати в цьому віртуальному кластері будь-яку контейнеризовану програму, забезпечуючи абстракції виявлення мережі/диска/сервісу, моніторинг життєвого циклу всіх контейнерів і перезапуск їх у разі виникнення проблем. Цей тип запуску є найвищим на даний момент, а масштабування визначається на рівні інтеграції програми – система сама дбає про те, щоб кожен екземпляр не був перевантажений, і одночасно збільшує кількість екземплярів програми, коли навантаження збільшується, і зменшується число, коли настане спад.

Програми та служби, які розширюють програми, можуть працювати разом, забезпечуючи: моніторинг, ведення журналів, планування виконання команд. Це можна розповсюдити в Service Mesh (рис. 6).

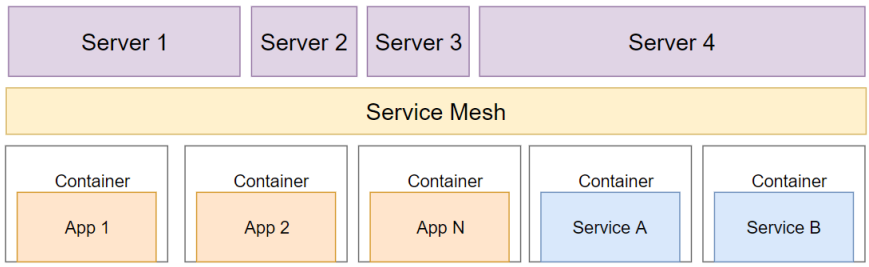

Рис. 6. Інтеграція додатків і сервісів у Service Mesh, група серверів утворює єдиний кластер, програми не взаємодіють із серверами безпосередньо

З іншого боку, вам потрібно контролювати розмір кластера для Service Mesh, якщо потужності цієї групи серверів недостатньо для масштабування самого кластера. Це завдання вирішується збільшенням кількості активних серверів або зміною розміру такої групи серверів, яка зазвичай простоює. У контексті додатків таке масштабування є вертикальним, у контексті кластера – вертикальним або горизонтальним.

Механізм балансування навантаження в базі даних. Після поглибленого аналізу методів розгортання веб-ресурсів, методів масштабування різних сервісів і способів підготовки проекту до масштабування, можна виділити такі порівняльні характеристики.

Оскільки файли не можуть бути у файловій системі програми для масштабування, ви повинні вибрати спосіб збереження таких файлів. У цій роботі пропонується комбінований метод масштабування баз даних без розробки рішення для кожного конкретного проекту з метою зниження витрат на розробку та підтримку таких проектів (табл.1).

Таблиця 1

Збереження користувацьких файлів під час запуску програми для забезпечення масштабованості

Метод	Інтеграція	Запис	Читання
GlusterFS / CephFS	Легко	Швидко	Між екземплярами можуть виникати затримки доставки
Amazon S3/Rackspace	Інтенсивно	Може виникнути затримка запису	Швидке читання з будь-якої точки світу
Спільний обсяг сервісної сітки	Легко	Швидко	Відносно швидко

Як результат порівняння – загальний обсяг Service Mesh має найбільше переваг, як це найкращий варіант для видалення спільних файлів користувача.

Вибір репозиторію для користувацьких сеансів можна представити таким порівнянням. Після поглибленого аналізу методів сеансів збереження – найуспішнішим є використання сховищ Key-Value (наприклад, Redis), оскільки вони зазвичай добре масштабуються (кластеризуються) і мають дуже хорошу швидкість читання/запису, зберігаючи дані безпосередньо в пам'яті. Єдиним недоліком є необхідність введення в експлуатацію нової служби та її обслуговування (табл.2).

Таблиця 2

Збереження користувацьких сеансів

Метод	Інтеграція	Запис	Читання	Масштабування
Збереження в реляційній базі даних (MySQL/PostgreSQL)	Легко	Повільно	Повільно	Забирає багато часу
Збереження в сховище ключів (Redis/Memcached)	Відносно легко	Швидко	Швидко	Легко

Способи розміщення web-додатка ви можете побачити в порівнянні в таблиці 3.

All rights reserved | Creative Commons Attribution 4.0 International License

1) Враховуючи ціну одиниці сервера, при реалізації великомасштабної архітектури на основі фізичних або виділених серверів вартість зростає в геометричній прогресії.

2) Деякі провайдери надають цю послугу, але зазвичай – можливість налаштування обмежена.

3) На даний момент Service Mesh має найвищу гнучкість серед будь-якого веб-ресурса, після налаштування він дозволяє запустити будь-який кластер за лічені секунди та вимкнути його. Хоча це рішення має середню вартість – але ця вартість дуже ефективна, оскільки кластер завжди має лише необхідну кількість ресурсів для своєї роботи, простою ресурсів немає.

4) Service Mesh побудований на ідеях високопродуктивних і високонадійних систем, тому автомасштабування є одним з основних механізмів, закладених у ньому.

Таблиця 3

Методи розміщення веб-ресурсів

Метод	Вартість	Легкість налаштування	Гнучкість	Автомасштабування
Фізичний / виділений сервер	Середня	Важко	Низька	Ні
Віртуальний сервер	Низька	Середня	Середня	Умовно відсутнє
Хмара	Середня	Просто	Висока	Так
Сервісна сітка	Середня	Середня	Найвища	Працює за замовчуванням

Результат цього порівняння показує, що якщо метою продукту є висока надійність, висока продуктивність і висока доступність, то найкращим вибором є Service Mesh, наприклад, Kubernetes. Це рішення дозволяє витратити час на налаштування лише один раз, а потім постійно користуватися перевагами гнучкої системи. Ще один позитивний ефект від використання цієї технології: всі додатки працюють в єдиному контейнерному режимі та автоматично переходять на принцип IoC (Infrastructure as a Code), що є останніми тенденціями на ринку.

Під час масштабування основної реляційної бази даних слід враховувати багато факторів, наприклад, характер навантаження від програми та обсяг даних. Можна зробити таке порівняння (табл.4).

Таблиця 4

Масштабування основної бази даних (MySQL/PostgreSQL)

Метод	Інтеграція	Простота використання	Відмовостійкість	Установка
Реплікація Master-Slave	Легка	Середня	Висока	Відносно легка
Майстер-Майстер реплікація	Легка	Висока	Висока	Дуже складна
Шардінг	Середня	Середня	Низька	Середня
Шардінг + Master-Slave	Середня	Середня	Висока	Важка

Після оцінки методів масштабування баз даних для досягнення найвищої стабільності та продуктивності найкращим вибором є «Шардінг + реплікація Master-Slave». У цього методу є недоліки: складність інтеграції та встановлення, але він дозволяє масштабувати базу даних до будь-якої кількості користувачів і підтримувати відмовостійкість.

Висновки. У цій статті пропонується вдосконалений механізм балансування навантаження між кількома екземплярами бази даних, що дозволяє розгортати будь-яку кількість екземплярів бази даних, щоб забезпечити швидкість продукту та відмовостійкість при високих навантаженнях. Використовуючи комбінований метод масштабування «Sharding + Master-Slave Replication», ви можете досягти потенційно необмеженого розміру бази даних без погіршення швидкості доступу до даних із можливістю самовідновлення у разі проблем. Service Mesh найкраще підходить для масштабування веб-ресурсів, оскільки він забезпечує необхідний запас гнучкості для задоволення потреб користувачів у довгостроковій перспективі.

Список використаних джерел:

[1] W. Iqbal, M. N. Dailey, D. Carrera. (2016) Unsupervised Learning of Dynamic Resource Provisioning Policies for Cloud-Hosted Multitier Web Applications. *IEEE Systems Journal* (10, №4), 1435-1446.

[2] О. І. Гороховський, Т. І. Трояновська. (2009) Моделі складових АСДН. *Вісник Хмельницького національного університету*. (3), 230–236.

[3] E. G. Radhika, G. Sudha Sadasivam, J. Fenila Naomi. (2018) An Efficient Predictive technique to Autoscale the Resources for Web applications in Private cloud. *Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB).* 27-28 February 2018, Chennai, India.

[4] С. М. Захарченко, Т. І. Трояновська, О. В. Бойко, В. С. Рибаченко (2016) Застосування односторінкових веб-орієнтованих інтерфейсів в соціально значущих проектах. *Вісник ХНУ* (3), 33-39.

[5] Q. Wang, H. Chen, S. Zhang, L. Hu,B. Palanisamy. (2019) Integrating Concurrency Control in n-Tier Application Scaling Management in the Cloud. *IEEE Transactions on Parallel and Distributed Systems* (30, №4), 855-869.

[6] Гороховський О. І., Трояновська Т. І., Азаров О. Д. (2016) *Інформаційна технологія доставки контенту у системах комп'ютеризованої підготовки спеціалістів*. Вінниця : ВНТУ.

[7] H. Chen, Q. Wang, B. Palanisamy, P. Xiong. (2017) DCM: Dynamic Concurrency Management for Scaling n-Tier Applications in Cloud. *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. 5-8 June 2017, Atlanta, Georgia, USA.

[8] О. І. Гороховський, Т. І. Трояновська. (2009) Інформаційна технологія розробки адаптивних дистанційних курсів. *Інформаційні технології та комп'ютерна інженерія* (2), 75–80.

[9] Olexiy D. Azarov, Tetiana I. Troianovska, Liudmyla A. Savytska, Tamara O. Savchuk, Larysa E. Nykyforova, Volodymyr A. Otryshko, Batyrbek Suleimenov, Konrad Gromaszek, Ainur Kozbekova, Azhan Sagymbekova. (2017) Quality of content delivery in computer specialists training system. *Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments* (10445). 7 August 2017, Wilga, Poland.

[10] Трояновська Т. І. (2013). Розробка протоколу за стандартом SCORM для обміну даними між складовими СКП. *Dynamika naukowych badan*. 7–15 липня 2013 р., Przemysl, Poland.

All rights reserved | Creative Commons Attribution 4.0 International License