УДК 004.72

А.В. Миргородський[1]
Л.Б. Ліщинська[1]

ЗАСТОСУВАННЯ КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ ПРИ РОЗРОБЦІ БАГАТОКОРИСТУВАЦЬКИХ ПРОГРАМНИХ СИСТЕМ

[1]Вінницький національний технічний університет

Анотація
Наведено короткі відомості про клієнт-серверну архітектуру, розглянуто шляхи вирішення типових проблем безпеки та масштабування при її застосуванні для розробки багатокористувацьких програмних систем.
Ключові слова: IT, клієнт-серверна архітектура, архітектура, багатокористувацькі системи.

Abstract
The paper provides brief information about the client-server architecture, considers ways to solve typical security and scaling issues when using it for the development of multi-user software systems.
Keywords: IT, client-server architecture, architecture, multi-user systems.

Вступ

Архітектура програмного забезпечення є одним з фундаментальних аспектів при проектуванні та розробці комплексних програмних систем, адже вона визначає її загальну організацію, базові принципи роботи та шляхи розширення, а також може накласти певні обмеження з точки зору функціональних та нефункціональних вимог. У сучасному світі переважна більшість програмних систем передбачає роботу з великою кількістю користувачів – це можуть бути як певні сервіси для віддаленої роботи, обміну повідомленнями в реальному часі та одночасного спільного редагування документів, так і платформи для дозвілля, як-от різноманітні медіасервіси, онлайн-ігри тощо.

Основними проблемами таких систем є захист доступу до ресурсів [1] та масштабування програмного забезпечення відповідно до навантаження [2]. Проблеми з доступом можуть вести до несанкціонованого розповсюдження інформації, а відповідно й до проблем з GDPR та інших питань з правової сфери. Неадаптивна до змін в навантаженні система не зможе в повній мірі задовільнити потреби кінцевих користувачів, що в свою чергу може нести прямі фінансові та репутаційні збитки для компанії. Застосування клієнт-серверної архітектури є популярним підходом при проектуванні багатокористувацьких систем, що може допомогти вирішити ці та інші труднощі.

Метою дослідження є вивчення переваг та недоліків клієнт-серверної архітектури при розробці багатокористувацьких програмних систем.

Об'єктом дослідження є процес розробки багатокористувацьких програмних систем.

Предмет дослідження – клієнт-серверна архітектура.

Основна частина

Клієнт-серверну архітектуру можна розглядати в якості прикладу розподіленої системи, що складається з двох основних компонентів – клієнта та сервера. Основне призначення сервера – це надавати необхідні сервіси та обслуговувати запити клієнта. Саме клієнт ініціює запити та обмін даними з сервером в той момент, коли йому необхідно звернутися до того чи іншого сервісу.

Фактична реалізація та складові компоненти клієнт-серверної архітектури можуть відрізнятися в залежності від потреб програмної системи. Зазвичай до характеристик цієї архітектури відносять [3]:

- чітке розподілення ролей – існує лише клієнт для створення запитів і сервер для їх обробки;
- строгий порядок обміну повідомленнями – сервер надсилає дані в відповідь на запит клієнта та самостійно не ініціює комунікацію, а дані, що відправляє клієнт, вважаються частиною його запиту;

• основна мета сервера – це зберігання інформації та її обмін з авторизованими клієнтами;

• клієнтів може бути декілька, вони не мають інформації одне про одного й можуть одночасно працювати з одним сервером.

Для комунікації між клієнтом і сервером часто застосовують протоколи TCP, UDP або ж похідні від них. Наприклад, переважна більшість веб-додатків та різноманітних онлайн-систем використовують протокол HTTPS для обміну повідомленнями. Деякі класичні концепції клієнт-серверної архітектури можуть змінюватися: сучасні веб-сервери з підтримкою HTTP/2 здатні самостійно надсилати дані клієнту без попереднього запиту, що може покращити швидкодію в деяких сценаріях використання. Також, різноманітні поліпшення на рівні протоколу, як-от повторне використання існуючого TCP-з'єднання, здатні збільшити продуктивність багатокористувацьких систем на основі клієнт-серверної архітектури шляхом оптимізації мережевої комунікації.

В деяких випадках клієнт та сервер можуть виступати в якості окремих програмних компонентів, що працюють на одному сервері або пристрої користувача. Це може застосовуватися для відділення бізнес-логіки окремих процесів програмної системи або ж ізоляції застосованих ресурсів. Прикладами такого підходу можуть бути система плагінів для IaC-інструменту Terraform або ж віконна система X Window System. З іншого боку, багатокористувацькі програмні системи рідко оперують в межах однієї машини та часто потребують більш розвиненої мережевої інфраструктури для забезпечення потрібного рівня швидкодії та безпеки.

Розглянемо більш просунутий приклад застосування клієнт-серверної архітектури для забезпечення роботи багатокористувацької програмної системи. На рисунку 1 наведено приклад з застосуванням сервісів хмарного провайдера AWS. Подібну архітектуру можна реалізувати за допомогою сервісів інших провайдерів або ж на основі програмного та апаратного забезпечення звичайних дата-центрів.

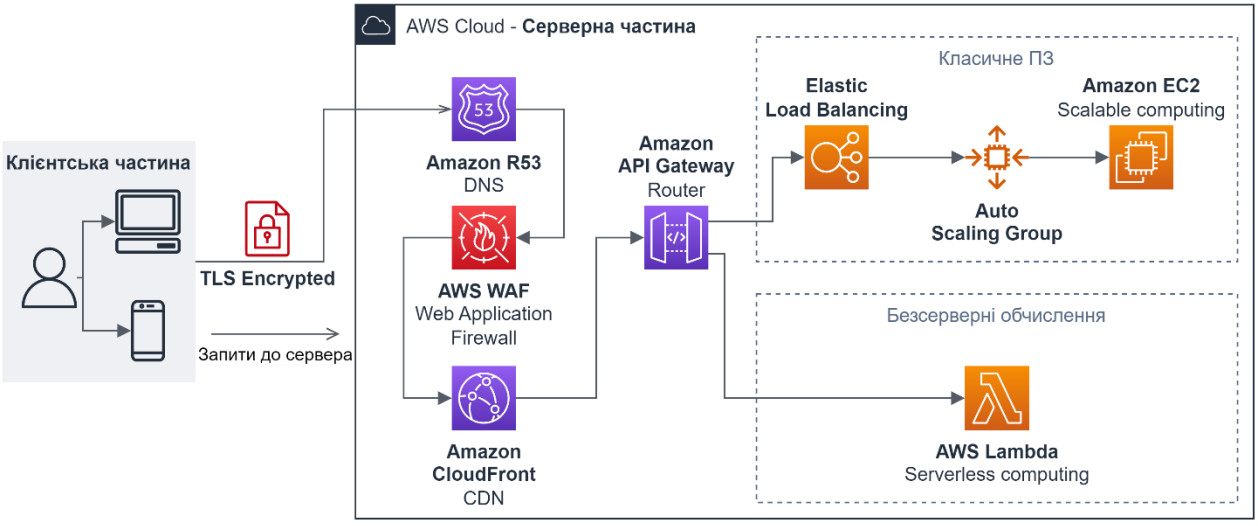

Рисунок 1 – Приклад просунутої клієнт-серверної архітектури на основі сервісів AWS

В якості клієнта для такої програмної системи може виступати будь-який пристрій, що здатен генерувати запити за допомогою попередньо визначеного протоколу зв'язку, а от серверна частина значно ускладняється. На її прикладі розглянемо основні переваги та недоліки клієнт-серверної архітектури, а також шляхи їх потенційного вирішення.

Першим важливим аспектом для багатокористувацьких систем є захист даних і клієнт-серверна архітектура не пропонує ніяких стандартних шляхів вирішення цього питання, а лише накладає ряд додаткових обмежень. Клієнтів може бути безліч, а шлях трафіку від користувацького пристрою до цільового сервера може включати в себе велику кількість неперевірених мережевих вузлів, тому при побудові архітектури варто розглядати протоколи обміну даними з підтримкою шифрування. Один з варіантів – це TLS-шифрування, що широко використовується в HTTPS та інших протоколах. Для покращення захисту в наведеному прикладі також застосовуються сервіси Amazon R53 та AWS WAF. R53 – це DNS сервіс, за допомогою якого можна гнучко налаштувати розподілення вхідного трафіку або ж заблокувати за певними критеріями (IP-адреса, геолокація тощо) вхідні запити, щоб на ранніх

етапах обробки відсікти потенційних зловмисників. Перевірка за допомогою DNS є грубим методом, тому для захисту від типових поширених загроз (ін'єкції коду, відомі вразливості ПЗ тощо) можна застосувати AWS WAF або подібні до нього сервіси та додатки, що сфокусовані на скануванні запитів і виявленні спроб застосування цих вразливостей. Наведений приклад є лише одним з можливих варіантів організації захисту в клієнт-серверній архітектурі.

Наступна проблема – це масштабування системи в залежності від навантаження. Клієнт-серверна архітектура передбачає централізоване зберігання і управління даними, тому така єдина точка входу для клієнтів може призвести до перевантаження сервера, деградації швидкодії або ж повної відмови системи. В розглянутому прикладі перший крок до вирішення цієї проблеми – це застосування CDN (англ. Content Delivery Network, мережа доставки контенту). Це дозволить зберігати в окремому кеші часто використовувані ресурси, що зменшить навантаження на програмну систему та збільшить швидкість обробки окремих запитів клієнтів. Від загрози повної відмови системи можна захиститися рядом додаткових методик, по типу горизонтального масштабування або ж високої доступності – вони спрямовані на розподілення навантаження між декількома серверами та дублюванні ресурсів, щоб уникнути проблем з єдиним монолітним сервером в класичній клієнт-серверній архітектурі. В наведеному прикладі в AWS для цього можна використати сервіси API Gateway для маршрутизації запитів та комбінацію ELB, ASG та EC2 для автоматичного масштабування. Альтернативою є застосування безсерверних розрахунків на основі AWS Lambda або сервісів-аналогів.

Висновок

Дослідження показало, що клієнт-серверна архітектура вирішує питання ролей та організації доступу до даних в багатокористувацьких системах, але залишає відкритими ряд питань та недоліків, по типу захисту даних та вразливості єдиної точки доступу для клієнтів. Розглянуті приклади та популярні підходи наводять варіанти вирішення цих проблем та подальшого покращення клієнт-серверної архітектури, що можуть бути використанні при розробці сучасних захищених і надійних багатокористувацьких програмних систем.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Dasgupta P., Wang Y. Security and reliability of client server systems on the internet. Tempe : Arizona State University, Bureau of Publications Tempe, 2006. 24 p.

2. Kuzmenko E. Client-Server Architecture. Advantages and Disadvantages of the Network Computing Model. KITRUM. URL: https://kitrum.com/blog/client-server-architecture-advantages-and-disadvantages/ (date of access: 19.05.2023).

3. Gautam S., Nandan K. Client Server Architecture. EnjoyAlgorithms. URL: https://www.enjoyalgorithms.com/blog/client-server-architecture (date of access: 19.05.2023).

Миргородський Андрій Вікторович – студент групи 3ПІ-22м, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: mirgorodskijav@gmail.com

Ліщинська Людмила Броніславівна – д-р техн. наук, професор кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: llb@vntu.edu.ua

***Myrhorodskyi Andrii* –** student of group 3PI-22m, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: mirgorodskijav@gmail.com

Lishchynska Lyudmyla Bronislavivna – Dr. Sc. (Eng.), Full Professor, Professor of Program Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: llb@vntu.edu.ua