УДК 004.75

А. С. Хом'як
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОБРОБКИ В РЕАЛЬНОМУ ЧАСІ У СЛУЖБАХ ЧАТ-БОТІВ ЧЕРЕЗ ІНТЕГРАЦІЮ ЧЕРГИ ЗАПИТІВ ДЛЯ РОЗПОДІЛЕННЯ НАВАНТАЖЕННЯ

A. Khomiak

REALTIME PROCESSING EFFICIENCY IMPROVEMENT IN CHATBOT SERVICES THROUGH THE USE OF REQUEST QUEUES FOR LOAD BALANCING

Зростаюча потреба у швидкому, ефективному обробленні природної мови у чат-ботах вимагає архітектури, здатної до швидкої відповіді та масштабованої продуктивності. Запропонована архітектура задовольняє ці вимоги, інтегруючи брокер повідомлень RabbitMQ для посередництва між API для генерації тексту та робочими вузлами з GPU.

Представлена система (зображена на рисунку 1) включає три ключові компоненти: API для генерації тексту, RabbitMQ як брокер повідомлень та робочі вузли, оснащені GPU. API для генерації тексту служить вхідною точкою для зовнішніх запитів. RabbitMQ забезпечує ефективне управління чергами повідомлень та їх розподілом. Робочі вузли, оснащені високопродуктивними GPU (Nvidia A100), керують обчислювально-інтенсивними завданнями генерування тексту.

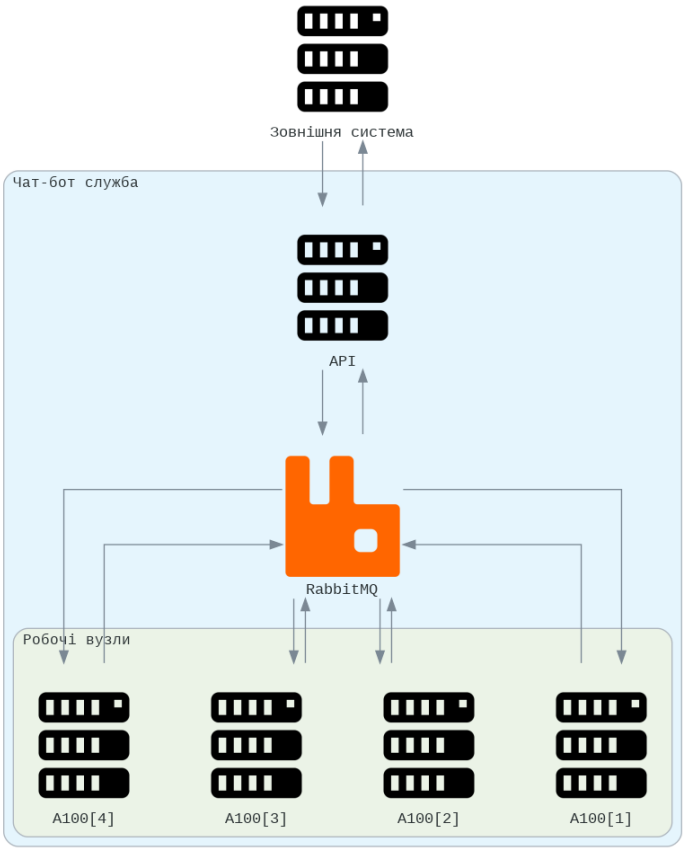

Рисунок 1. Запропонована архітектура чат-бот служби

RabbitMQ є основою нашої архітектури, що забезпечує асинхронне спілкування між API та робочими вузлами, тим самим розділяючи ці компоненти. Таке рішення дозволяє безперервно масштабувати та керувати змінними навантаженнями без зниження продуктивності. Після отримання запиту, API передає його до спеціально визначеної черги RabbitMQ. Цей механізм зменшує затримку API, оскільки він уникає необхідності синхронної обробки робочими вузлами, тим самим полегшуючи швидку обробку вхідних запитів.

Кожен робочий вузол в цій архітектурі оснащений GPU, спеціально призначеними для завдань глибокого навчання. Ці GPU значно прискорюють обробку моделей природної мови, що є критичним фактором для чат-ботів в реальному часі. Робочі вузли постійно слідкують за чергою RabbitMQ на предмет вхідних запитів. Після отримання запиту, робочий вузол обробляє його та передає результат назад до API через чергу відповідей в RabbitMQ.

Однією з помітних переваг цієї архітектури є її вроджена масштабованість. RabbitMQ ефективно розподіляє завдання між робочими вузлами, дозволяючи збалансоване розподілення навантаження. Це, разом з обчислювальною потужністю GPU, дозволяє системі масштабуватися для реагування на зростаючі обсяги запитів без компромісів у часі відповіді. Простота масштабування досягається за рахунок відсутності потреби в зміні конфігурації при зміні кількості робочих вузлів, чого не можна досягнути використовуючи підходи на основі реверс-проксі та традиційні розподілювачі навантаження без автоматичного виявлення служб (service discovery), що може бути неможливим у випадку гібридних розгортувань. Це, зокрема, дозволяє використовувати обчислювальні потужності, які можуть бути економічно вигідними однак географічно віддаленими від основних компонентів системи.

Література

1. "Distributing messages using RabbitMQ with advanced message exchanges," Int. J. Res. Stud. Comput. Sci. Eng., vol. 6, no. 2, 2019.
2. S. T and S. N. K, "A study on Modern Messaging Systems- Kafka, RabbitMQ and NATS Streaming," CoRR, vol. abs/1912.03715, 2019. [Online]. Available: http://arxiv.org/abs/1912.03715
3. J. Dizdarevic, M. Michalke, and A. Jukan, "Engineering and Experimentally Benchmarking Open Source MQTT Broker Implementations," 2023. [Online]. Available: arXiv:2305.13893