МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ
VI ФОРУМУ
«Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології»
AERT-2024

6 - 7 грудня 2024 р.

Харків 2024

EDGE-ОБЧИСЛЕННЯ ТА ЇХ РОЛЬ У МОБІЛЬНИХ ЗАСТОСУНКАХ: РОЗРОБКА НА JAVA

ст. викл каф. МТС, Чумак В.С., ас. Бойко Н.В.
Харківський національний університет радіоелектроніки,
кафедра мікропроцесорних технологій і систем, м. Харків, Україна
e-mail: valeriia.chumak@nure.ua

Abstract. This article examines modern approaches to implementing edge computing in the context of mobile applications on the Java platform. The research focuses on methods for optimizing data processing on edge devices, reducing latency, and minimizing energy consumption. Practical examples of implementing edge computing patterns in Java are presented, and their efficiency is analyzed based on experimental data. The results demonstrate a significant improvement in application performance when applying the proposed approaches.

Key words. Edge Computing, Mobile Edge Computing (MEC), Edge Processing, Low Latency Computing, Distributed Computing, Mobile Development, Java, Android.

Introduction. Edge-обчислення представляють собою парадигму розподілених обчислень, яка наближає обробку даних до джерела їх генерації. Впровадження edge-обчислень у мобільні застосунки дозволяє суттєво знизити затримку обробки даних порівняно з традиційними хмарними рішеннями завдяки локальній обробці безпосередньо на пристрої. Стандартні сервери або хмарні сервіси часто не можуть забезпечити необхідну швидкість обробки даних через високі затримки, що виникають при передаванні великих обсягів інформації між користувачем і сервером. Edge-обчислення дозволяє переносити обробку даних безпосередньо до кінцевих пристроїв або на локальні сервери, тим самим зменшуючи затримки та збільшуючи швидкість реакції застосунків. У контексті мобільних застосунків це особливо важливо для таких сценаріїв, як обробка відео, розпізнавання зображень, ігри з розширеною реальністю (AR), а також для застосунків, що працюють в реальному часі, таких як моніторинг здоров'я.

Main body. Edge-обчислення стає особливо корисним у мобільних застосунках, де є обмеження за швидкістю передачі даних та обчислювальними ресурсами. Основні переваги:

- зниження затримок: обробка даних безпосередньо на пристрої або в локальних серверах дозволяє значно зменшити час відгуку, оскільки дані не передаються через Інтернет до віддалених серверів. Для застосунків, які потребують високої швидкості обробки, таких як ігри або відеоаналітика, це критично важливо.

- економія пропускної здатності: оскільки лише частина даних передається в мережу, а більшість обробляється локально, знижується навантаження на канали зв'язку і зменшуються витрати на трафік.

- безпека та конфіденційність: в edge-обчисленнях чутливі дані можуть бути оброблені безпосередньо на пристрої, що знижує ризики їх витоку при передачі в мережу. Це особливо важливо для медичних і фінансових застосунків, де конфіденційність даних є критичним аспектом.

Розробка мобільних застосунків з використанням edge-обчислень на Java вимагає врахування ресурсів пристроїв. навіть з обмеженими ресурсами, можливо реалізувати ефективну обробку за допомогою оптимізації алгоритмів, таких як використання паралельних потоків (multi-threading), асинхронних обчислень, а також покращених структур даних. Також для покращення продуктивності обробки на мобільних пристроях можна використовувати апаратне прискорення, яке підтримується більшістю сучасних смартфонів. Наприклад, можна використовувати GPU або інші спеціалізовані процесори для пришвидшення обчислень в таких сценаріях, як обробка зображень або відео. Для інтеграції edge-обчислень у мобільні застосунки на Java можна використовувати такі API, як Android Neural Networks API (NNAPI) для роботи з нейронними мережами, або TensorFlow Lite для роботи з вбудованими моделями машинного навчання на мобільних пристроях.

Основний патерн edge-обчислень у мобільних застосунках – локальна передобробка даних перед їх відправленням у хмару. Цей підхід має кілька ключових переваг, т.я. зменшення мережевого трафіку: за даними досліджень, локальна агрегація даних дозволяє суттєво скоротити обсяг переданих даних, що особливо важливо для мобільних пристроїв з обмеженим трафіком, покращення приватності: попередня обробка даних на пристрої дозволяє фільтрувати конфіденційну інформацію перед відправленням у хмару, що підвищує загальний рівень безпеки застосунку, оптимізація батареї: пакетна обробка даних (batching) дозволяє зменшити кількість мережевих з'єднань, що позитивно впливає на споживання енергії. Реалізація:

```
public class EdgeDataProcessor {
    private static final int BATCH_SIZE = 1000;
    private final Queue<SensorData> dataQueue;
    public EdgeDataProcessor() {
        this.dataQueue = new ConcurrentLinkedQueue<>();
    }
    public void processSensorData(SensorData data) {
        // Локальна фільтрація та агрегація
        if (isRelevantData(data)) {
            dataQueue.offer(data);
            // Обробка батчами для оптимізації
            if (dataQueue.size() >= BATCH_SIZE) {
                List<SensorData> batch = new ArrayList<>();
                for (int i = 0; i < BATCH_SIZE; i++) {
```

```
                batch.add(dataQueue.poll());
            }
            processDataBatch(batch);
        }
    }
}
private void processDataBatch(List<SensorData> batch) {
    // Агрегація даних
    AggregatedData aggregated = batch.stream()
        .collect(Collectors.groupingBy(
            SensorData::getSensorType,
            Collectors.averagingDouble(SensorData::getValue)
        ));
    // Відправка агрегованих даних у хмару
    CloudService.sendData(aggregated);
}
}
```

Результати. Використання ConcurrentLinkedQueue забезпечує потокобезпечну обробку даних без блокування всього застосунку. Це особливо важливо для мобільних пристроїв, де ресурси обмежені. Реалізований механізм батчингу (BATCH_SIZE = 1000) дозволяє зменшити кількість мережевих запитів оптимізувати використання CPU через групову обробку, знизити навантаження на батарею. Метод isRelevantData() дозволяє відсіювати нерелевантні дані ще на етапі збору, що зменшує навантаження на подальшу обробку. Отже маємо зниження мережевого трафіку порівняно з прямою відправкою даних для типових сценаріїв використання, зменшення споживання батареї завдяки оптимізованій пакетній обробці, покращення часу відгуку застосунку через зменшення кількості мережевих операцій. Такий підхід особливо ефективний для IoT застосунків та систем моніторингу, де потрібна обробка великої кількості сенсорних даних.

Реальні сценарії застосування edge-обчислень на мобільних пристроях включають: медичні застосунки для аналізу біометричних даних, таких як серцевий ритм або рівень кисню в крові, обробка цих даних на пристрої дозволяє забезпечити миттєвий доступ до результатів без необхідності передавати їх на сервер, ігри з доповненою реальністю (AR), у застосунках для відеоспостереження.

Попри численні переваги, використання edge-обчислень в мобільних застосунках з'являються певні проблеми. Мобільні пристрої мають менше обчислювальних ресурсів порівняно з серверними інфраструктурами. Це обмежує можливості для масштабних обчислень. Також для деяких застосунків потрібно здійснювати синхронізацію між локальними і віддаленими даними, що може бути складним завданням.

Conclusion. Edge-обчислення відкривають нові можливості для покращення продуктивності та зменшення затримок у мобільних застосунках. Використання Java для розробки таких рішень дозволяє зберегти ефективність і забезпечити високу продуктивність навіть при обмежених ресурсах мобільних пристроїв. Реалізація таких рішень вимагає використання специфічних інструментів і оптимізації коду, однак в результаті користувачі отримують значно кращий досвід взаємодії з застосунками, що працюють у реальному часі. Подальші дослідження можуть бути спрямовані на розробку більш складних алгоритмів розподілу обчислювального навантаження між edge-пристроями та хмарою, а також на оптимізацію енергоспоживання в умовах обмежених ресурсів.

References.

1. Official Android Developer Documentation. Edge Computing and Local Processing [Електронний ресурс]. – Режим доступу: https://developer.android.com/ – Дата доступу: 20 листопада 2024 р.
2. Java Documentation. Concurrent Programming in Java [Електронний ресурс]. – Режим доступу: https://docs.oracle.com/javase/tutorial/essential/concurrency/ – Дата доступу: 20 листопада 2024 р.
3. Android Developers Guide. Background Processing Best Practices [Електронний ресурс]. – Режим доступу: https://developer.android.com/guide/background – Дата доступу: **20** листопада 2024 р.
4. Аналіз принципів побудови телемедичних комплексів широкого призначення / В. С. Чумак, О. Г. Аврунін, Є. А.Чугуй, І. В. Свид // АСУ та прилади автоматики. 2021. № 177. С. 80-85.
5. Malka, M., Farhan, E., Morgenstern, H., & Shlezinger, N. Decentralized low-latency collaborative inference via ensembles on the edge // IEEE Transactions on Wireless Communications. – 2024.
6. Mao, Yuyi, et al. A survey on mobile edge computing: The communication perspective // IEEE Communications Surveys & Tutorials. – 2017. – Т. 19, № 4. – С. 2322–2358.
7. Чумак В. С. Інтеграція нейронних мереж у медичні пристрої на основі STM32 для автоматичної діагностики та моніторингу пацієнтів / В. С. Чумак // Автоматизація, електроніка та робототехніка. Стратегії розвитку та інноваційні технології (AERT-2023) : матеріали V форуму, 29–30 листопада 2023 р. – Харків : ХНУРЕ, 2023. – С. 132-133.
8. Mach, Pavel, Becvar, Zdenek. Mobile edge computing: A survey on architecture and computation offloading // IEEE Communications Surveys & Tutorials. – 2017. – Т. 19, № 3. – С. 1628–1656.