

ПРОЄКТУВАННЯ АРХІТЕКТУРИ ВИСОКОНАВАНТАЖЕНОЇ РОЗПОДІЛЕНОЇ СИСТЕМИ З ОБМЕЖЕНИМ ЧАСОМ ОБРОБЛЕННЯ ЗАПИТІВ

Козкін Євгеній Ігорович

студент групи МІПМ-1-26-1.4 д.,

Київський столичний університет імені Бориса Грінченка, м. Київ,

науковий керівник — к.т.н., доц. Машкіна І.В.

ВСТУП

Актуальність і постановка проблеми

У системах рекламних торгів у реальному часі важливими є як коректність результату, так і час його отримання. Специфікація OpenRTB визначає структуру запитів і відповідей учасників торгів. Поле *tmax* задає максимальний час отримання ставок у мілісекундах, включно з інтернет-затримкою [1]. Тому продуктивність RTB-системи потрібно оцінювати не лише за кількістю завершених операцій, а й за дотриманням часових обмежень.

Оброблення одного запиту може охоплювати перевірку даних, звернення до інших сервісів і формування підсумкової відповіді. Наприклад, Prebid Server перевіряє вхідні запити, викликає адаптери партнерів і формує відповідь після отримання результатів або завершення часу очікування [2]. Для запланованого прототипу ця послідовність слугуватиме орієнтиром під час визначення відповідальності компонентів.

За перевантаження взаємозалежних сервісів можуть зростати затримки та виникати відмови, що розглядається, зокрема, у дослідженні Rajomon [3]. Для реалізації мовою Go додаткове значення мають витрати на виділення пам'яті та її обслуговування збирачем сміття [5]. Отже, пошук причин уповільнення потребуватиме вимірювання часу виконання, використання ресурсів і поведінки компонентів, а не лише зміни одного загального налаштування.

Проблема цієї роботи полягатиме у виборі та перевірці архітектурних рішень, які дозволятимуть ефективніше обробляти RTB-запити за заданого ресурсного бюджету. Конкретний компонент для оптимізації буде визначено після реалізації та профілювання базового прототипу. Це дозволить пов'язати зміни з виявленим обмеженням продуктивності, а не наперед припускати, що певний механізм обов'язково забезпечить покращення.

Дослідження буде зосереджене на серверному обробленні запитів. Економічні стратегії торгів, прогнозування кліків і максимізація рекламного доходу не входитимуть до його предмета.

Мета, об'єкт, предмет і завдання дослідження

Мета дослідження — підвищення ресурсної ефективності та своєчасності оброблення RTB-запитів шляхом проєктування архітектури розподіленої системи, виявлення обмежень продуктивності та експериментального обґрунтування способів їх усунення.

Об'єкт дослідження — процес оброблення рекламних запитів у високонавантажній розподіленій RTB-системі за часових і ресурсних обмежень.

Предмет дослідження — архітектурні рішення та методи оптимізації оброблення RTB-запитів, а також їхній вплив на використання пам'яті, пропускну здатність і своєчасність отримання результатів за заданих часових і ресурсних обмежень.

Завдання дослідження: проаналізувати сучасні підходи; визначити вимоги до RTB-прототипу; реалізувати базовий сценарій; виконати профілювання; обрати та перевірити одну цільову зміну; порівняти варіанти за продуктивністю, ресурсними витратами та дотриманням дедлайнів.

Аналіз останніх досліджень і публікацій

Специфікація OpenRTB [1] буде використана для визначення підтримуваного формату взаємодії. Документація Prebid Server [2] слугуватиме прикладом організації серверного збирання ставок. Прототип охоплюватиме лише необхідний для експерименту сценарій, а не всі можливості цих рішень.

У роботі Rajomom запропоновано узгоджене керування перевантаженням у графі мікросервісів. Автори оцінюють корисну пропускну здатність і хвостові затримки за зміни навантаження [3]. Для запланованого дослідження важливим буде принцип спільного оцінювання кількості виконаної роботи та її своєчасності. Конкретний алгоритм Rajomom не планується відтворювати як обов'язкову частину прототипу.

У дослідженні SIEVE розглянуто ефективність кешування разом із пропускну здатністю та складністю реалізації політики витіснення [4]. Це приклад того, що оцінювання окремого механізму має враховувати його користь і накладні витрати. Звернення до цього джерела не означатиме обов'язкового впровадження SIEVE або проведення окремого дослідження алгоритмів кешування.

Офіційний посібник Go з керування пам'яттю пояснює залежність витрат збирача сміття від інтенсивності алокацій, обсягу живих даних і структури об'єктів [5]. Засоби діагностики Go дозволяють досліджувати процесорний час, пам'ять і блокування [6]. Ці джерела будуть використані для інтерпретації вимірювань конкретної реалізації.

РЕЗУЛЬТАТИ ПОЧАТКОВОГО ЕТАПУ ДОСЛІДЖЕННЯ

Робота перебуває на стадії постановки задачі та проектування експерименту. Отриманими на цьому етапі результатами є окреслення меж прототипу, визначення вимог до коректності, вибір засобів розроблення та формування плану вимірювань. Реалізація прототипу та порівняльні експерименти заплановані. На поточному етапі кількісні результати відсутні, тому переваги окремого механізму не стверджуються.

Архітектура та межі прототипу

Запланований прототип складатиметься з генератора навантаження, HTTP API оркестратора, емуляторів джерел ставок та модуля формування відповіді. Потік оброблення: отримання й перевірка OpenRTB-запиту → конкурентні звернення до джерел → отримання відповідей у межах дедлайну → вибір допустимої ставки → формування результату [1, 2].

Для порівнюваності експериментів планується зафіксувати один рекламний показ у запиті, один формат реклами, одну валюту та однакове

правило вибору найвищої допустимої ставки. Тестові дані завантажуватимуться з підготовлених JSON-файлів. Усі варіанти реалізації працюватимуть за однакових правил відбору; спрощені правила будуть умовами стенду, а не твердженням про всі рекламні платформи.

Емулятори джерел дозволятимуть відтворювати звичайні відповіді, затримки, технічні помилки та no-bid. Реальні комерційні інтеграції не будуть необхідною умовою виконання роботи. Білінг, рекламний кабінет і автоматичне масштабування не входитимуть до обов'язкового обсягу.

Базовий прототип одразу матиме перевірку запитів, часові обмеження, контроль кількості одночасних операцій і скасування простроченої роботи. Точний часовий бюджет стенду буде задано до початку вимірювань. Для відповіді враховуватимуться вже витрачений час і резерв завершення, без повторного врахування тих самих мережових витрат.

Гіпотези та підхід до оптимізації

Попередня гіпотеза Н1: основне обмеження продуктивності залежатиме від характеру навантаження і може переміщуватися між мережевими викликами, обробленням даних та використанням пам'яті. Гіпотеза Н2: цільова зміна, обрана після профілювання, зможе покращити один або кілька показників, але її ефект залежатиме від сценарію та додаткових накладних витрат. Гіпотеза Н3: оцінювання лише середнього часу відповіді не буде достатнім для встановлення придатності системи до роботи за жорсткого дедлайну. Ці припущення потребують перевірки, а не подаються як встановлені результати.

Планується перевіряти одну цільову зміну за раз. Для кожного порівняння будуть зафіксовані проблема, очікуваний ефект, можливі додаткові витрати та показники оцінювання. Якщо застосовуватиметься кешування, до перевірки входить актуальність результатів; якщо змінюватиметься організація мережових викликів — дотримання часових обмежень і завершення роботи після скасування.

Технології та план реалізації

Прототип планується реалізувати мовою Go з використанням net/http для мережевої взаємодії та context для передавання дедлайнів і сигналів скасування. Прикладна логіка буде відокремлена від HTTP-обробників, щоб її можна було тестувати незалежно від мережі. Базовий і альтернативний варіанти матимуть однаковий інтерфейс.

Для локального розгортання використовуватиметься Docker Compose [9], для функціональних перевірок — Go testing [7], для дослідження CPU, пам'яті та блокувань — rrprof [6], а для навантажувальних випробувань — k6 [8]. Параметри середовища та версії залежностей будуть зафіксовані. Дедлайни і скасування операцій реалізовуватимуться засобами пакета context [10].

Початковий стенд працюватиме на одному комп'ютері. Параметри контейнерів, версії програмного забезпечення й характеристики обладнання будуть зафіксовані. Результати характеризуватимуть саме цей стенд і не подаватимуться як підтвердження масштабованості на незалежних фізичних вузлах.

Методика експериментальної перевірки

Основне порівняння планується виконувати за однакових даних, правил формування відповіді, ресурсних обмежень і характеристик джерел ставок. Сценарії включатимуть стале навантаження, його зростання до перевантаження та короткочасний сплеск із поверненням до початкової інтенсивності. Параметри тестових даних і затримок будуть явно зазначені.

Оцінюватимуться час відповіді, зокрема 95-й і 99-й перцентилі, пропускна здатність, частка коректних відповідей у межах дедлайну, відхилення й технічні помилки. До ресурсних показників входимуть використання RAM і CPU, а для досліджуваних операцій — обсяг і кількість виділень пам'яті. Пам'ять Go-купи та всього процесу обліковуватиметься окремо.

Відсутність ставки no-bid є передбаченим протоколом результатом [1]. Тому вона відокремлюватиметься від технічних відхилень і перевищення часу очікування. Додатково враховуватиметься кількість своєчасно отриманих валідних ставок відносно всього вхідного потоку. Це допоможе виявити ситуації, коли зменшення затримки досягається через пропуск потрібних звернень або необґрунтоване повернення порожніх результатів.

Кожне порівняння передбачатиме повторні запуски з відокремленням підготовки від основного вимірювання. Для налаштувань, обраних за попередніми тестами, буде виконано додаткові перевірки на іншому наборі запитів. Планується показувати розкид результатів і зіставляти виграш із додатковими витратами, а не подавати лише найкраще значення.

Передбачається випробувати стале навантаження, поступове перевантаження і короткочасні сплески. Профілювальні запуски буде відокремлено від контрольних вимірювань, щоб мінімізувати накладні витрати діагностики [6? 11].

Очікувані результати

Планується отримати експериментальний прототип RTB-системи, базовий і змінений варіанти досліджуваного компонента та відтворювану методику їх порівняння. Профілювання має допомогти встановити основні джерела витрат, а навантажувальні випробування — перевірити вплив обраної зміни на систему загалом.

Очікуваним практичним результатом будуть рекомендації щодо застосування перевіреного рішення за визначених обмежень пам'яті, процесорного ресурсу та часу відповіді. У рекомендаціях планується відобразити не лише сприятливі сценарії, а й умови відсутності виграшу або появи додаткових витрат.

Результати майбутніх випробувань буде подано в порівняльній таблиці та, за потреби, на графіках залежності показників від навантаження. На поточному етапі вимірюваних значень немає, тому кількісне покращення, найкращий механізм і його параметри не заявляються.

ВИСНОВКИ

На початковому етапі визначено предметні межі дослідження, вимоги до RTB-прототипу, інструменти розроблення та показники оцінювання.

Запропоновано послідовність «базова реалізація — профілювання — цільова зміна — повторна перевірка». Обраний механізм оптимізації та його ефективність буде встановлено лише після експериментів. Результати одновузлового стенду не трактуватимуться як доказ масштабованості на незалежних фізичних вузлах.

ДЖЕРЕЛА

1. IAB Tech Lab. OpenRTB 2.6 Specification. 2022. URL: <https://iabtechlab.com/wp-content/uploads/2022/05/OpenRTB-2.6.pdf>
2. Prebid.org. Prebid Server: OpenRTB2 Auction Endpoint. URL: <https://docs.prebid.org/prebid-server/endpoints/openrtb2/pbs-endpoint-auction>
3. Xing J., Giannoukos A., Loh P., Wang S., Qiu J., Demoulin H. M., Kallas K., Lee B. C. Rajomon: Decentralized and Coordinated Overload Control for Latency-Sensitive Microservices. 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25). USENIX Association, 2025. P. 21–36. URL: <https://www.usenix.org/conference/nsdi25/presentation/xing>
4. Yang J., Zhang Y., Yue Y., Vigfusson Y., Vinayak R. SIEVE: Cache Eviction Can Be Simple, Effective, and Scalable. USENIX ;login:, 2024. URL: <https://www.usenix.org/publications/loginonline/sieve-cache-eviction-can-be-simple-effective-and-scalable>
5. The Go Authors. A Guide to the Go Garbage Collector. URL: <https://go.dev/doc/gc-guide>
6. The Go Authors. Diagnostics. URL: <https://go.dev/doc/diagnostics>
7. The Go Authors. Package testing. URL: <https://pkg.go.dev/testing>
8. Grafana Labs. Open and Closed Models; Constant Arrival Rate. URL: <https://grafana.com/docs/k6/latest/using-k6/scenarios/concepts/open-vs-closed/>
9. Docker Inc. Docker Compose Documentation. URL: <https://docs.docker.com/compose/>
10. The Go Authors. Package context. URL: <https://pkg.go.dev/context>
11. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., Hlushak, O., Zhylytsov, O., Ilich, L., Kobets, N., Kovaliuk, T., Kuchakovska, H., Lytvyn, O., Lytvyn, P., Mashkina, I., Morze, N., Nosenko, T., Proshkin, V., Radchenko, S., & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science. URL: <https://doi.org/10.28925/9720213284km>