УДК 004.451:004.738.5:004.75
В.А. Яцишин, П.В. Налутка, О.В. Доберчак, І.О. Боднарчук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

СУЧАСНА АРХІТЕКТУРА СТРІМІНГОВИХ ПЛАТФОРМ В РЕАЛЬНОМУ ЧАСІ

V.A. Yatsyshyn, P.V. Nalutka, O.V. Doberchak, I.O. Bodnarchuk

MODERN ARCHITECTURE OF REAL-TIME STREAMING PLATFORMS

На відміну від систем минулих часів, які спиралися на асинхронну та пакетну обробку, архітектура програмного забезпечення реального часу зараз є важливою в сучасному цифровому світі, де миттєва обробка інформації є нормою. Швидкість, надійність та передбачуваність є ключовими в застосунках реального часу, від платформ електронної комерції, що реагують на дії клієнтів, до фінансових систем, що виконують транзакції за мікросекунди. Саме тут на допомогу приходить архітектура програмного забезпечення реального часу, яка робить рівень продуктивності цих застосунків прийнятним для роботи в режимі реального часу.

Термін «реальний час» означає для системи, що вона повинна реагувати в чітко встановлені терміни або дедлайни. Для архітекторів та розробників розуміння того, як проектувати ці системи, є критично важливим для створення рішень у відповідності од сформульованих вимог. Навіть правильний результат може бути марним, якщо він надходить занадто пізно.

Системи реального часу поділяються на три категорії:

- Системи жорсткого реального часу. Недотримання термінів є системним збоєм. Прикладами є промислові системи управління та торговельні платформи, де час транзакцій є критично важливим.
- Системи роботи в режимі реального часу надійних систем. Пропуск термінів погіршує якість обслуговування, але не призводить до збою системи. Прикладами є відеоконференції, де випадкові переривання кадрів дратують, але не завершують дзвінок.
- Системи м’якого реального часу. Корисність результату знижується після закінчення терміну, але система продовжує функціонувати. Прикладами є механізми рекомендацій контенту, де персоналізація з невеликою затримкою все ще цінна.

Щоб врахувати ці різні очікування, цільовий додаток має бути спроектований та розроблений з урахуванням продуктивності в режимі реального часу. Незалежно від очікуваних термінів отримання результатів, добре побудована архітектура реального часу керує ресурсами, плануванням завдань, зв'язком та обробкою помилок, щоб забезпечити дотримання часових обмежень для своєї конкретної категорії.

Також важливо розуміти, що підпадає під архітектуру програмного забезпечення реального часу, а що ні (таблиця 1). Пакетна обробка, яка опрацьовує дані масово через заплановані проміжки часу, є класичним прикладом системи, що не працює в реальному часі. Хоча архітектура реального часу є поширеним шляхом, не всі варіанти використання та сценарії вимагають таких можливостей.

Майже всі програми містять компоненти реального часу поряд з аналізом історичних даних. Цей зсув підкреслює важливу роль програмного забезпечення та архітектур даних реального часу в сучасних програмах, що зумовлено ключовими факторами, серед яких варто згадати наступні.

Операції, критично важливі для бізнесу.

Для багатьох систем час впливає на бізнес-результати. Численні програми та галузі покладаються на справжні системи реального часу, щоб забезпечити дохід. Ось деякі приклади цього:

- Платформи електронної комерції: оновлення товарних запасів у режимі реального часу, персоналізація та обробка транзакцій безпосередньо впливають на коефіцієнти конверсії та задоволеність клієнтів.
- Фінансові послуги: торговельні платформи, системи обробки платежів та виявлення шахрайства потребують для роботи часу реагування на рівні мілісекунд.

Таблиця 1. Порівняння програмної архітектури реального часу та решти

Аспект	**Архітектура програмного забезпечення реального часу**	**Архітектура програмного забезпечення не для роботи в реальному часі**
випадки використання	Виявлення шахрайства, персоналізація в режимі реального часу, моніторинг у реальному часі, торгові додатки	Нарахування заробітної плати, створення звітів, пакетний імпорт, публікація контенту
Критерії успіху	Залежить як від точності *, так і* від своєчасності результатів	Залежить виключно від точності, незалежно від того, коли буде отримано результат
Дизайн застосунку	Компоненти, що реагують на події, неблокуючі та враховують час	Синхронні, запит/відповідь, блокувальні потоки прийнятні
Структура коду	Пріоритетність передбачуваних шляхів виконання, мінімальний вплив на збирання сміття (GC), асинхронний ввід/вивід	Надає пріоритет ремонтопридатності або пропускній здатності над точністю часу
Вплив технічного боргу	Архітектурно-технічний борг створює затримки, непередбачуваність та пропущені терміни — катастрофічні наслідки для систем реального часу. Навіть незначні недоліки, такі як блокування викликів або необмежені черги, можуть порушувати угоди про рівень обслуговування (SLA) або спричиняти каскадні збої.	Борг уповільнює доставку, збільшує витрати на обслуговування та може погіршити продуктивність, але рідко призводить до негайного збою. Терміни є гнучкими.

Кращий користувацький досвід.

Як ми вже обговорювали, очікування «миттєвого» обслуговування є складним завданням. Справжнього миттєвого зворотного зв'язку можна досягти лише завдяки інтеграції можливостей реального часу в базові сервіси. Наприклад, користувачі очікують миттєвого зворотного зв'язку, використовуючи:

- Веб- та мобільні додатки: адаптивні інтерфейси із часом завантаження менше секунди та миттєвими оновленнями (наприклад, стрічки соціальних мереж, спільне редагування) зараз є нормою [3].
- Потокові сервіси: доставка контенту з мінімальною буферизацією та адаптивною якістю вимагає прийняття рішень у режимі реального часу.

Прийняття рішень на основі даних.

У застарілих системах підприємствам іноді доводилося чекати годинами або навіть днями, поки великі пакети даних будуть оброблені та нададуть аналітику. Тепер, покладаючись на такий підхід, ви значно відстанете від конкурентів. Ось чому підприємства використовують аналітику в режимі реального часу для миттєвого отримання інформації, такої як:

- Платформи для взаємодії з клієнтами: аналіз поведінки користувачів у режимі реального часу дозволяє динамічну персоналізацію та цілеспрямоване втручання.
- Бізнес-аналітика: інформаційні панелі з візуалізацією даних у реальному часі дозволяють негайно реагувати на зміну умов.

Системи, керовані подіями.

Відбувся масовий зсув у бік подієво-орієнтованих систем та архітектур. У цих випадках архітектура реального часу є основним компонентом, який забезпечує функціонування всієї системи. Сучасні розподілені системи часто покладаються на обробку подій у реальному часі:

- Мікросервіси [1]: зв'язок між сервісами, керований подіями, вимагає своєчасної доставки та обробки повідомлень.
- IoT-додатки: обробка потоків даних датчиків у режимі реального часу забезпечує швидку автоматизацію та моніторинг.

Щоразу, коли час впливає на бізнес-цінність, задоволення користувачів або операційну ефективність, необхідна архітектура програмного забезпечення реального часу. Розглянемо основні принципи, які втілюють потреби бізнесу при впровадженні систем реального часу.

1. Своєчасність та передбачуваність.

Найважливішим елементом є те, що система повинна гарантувати виконання завдань у встановлені терміни. Це означає передбачувані алгоритми, обмежені шляхи виконання та відповідну пріоритезацію подій. Наприклад, служба обробки платежів повинна перевіряти, обробляти та підтверджувати транзакції протягом мілісекунд, щоб підтримувати пропускну здатність у періоди пікового навантаження.

2. Управління ресурсами.

Щоб дотримуватися цих термінів, системні ресурси необхідно розподіляти ефективно, щоб запобігти конфліктам, які можуть призвести до пропуску термінів. Це означає зосередження на:

- Керування пам'яттю з мінімальними паузами збирання сміття.
- Планування процесора, яке надає пріоритет критично важливим операціям.
- Розподіл пропускної здатності мережі для критично важливих потоків даних.

3. Контроль паралельності.

Багато систем реального часу обробляють безперервні масивні операції читання та запису, що вимагає ефективного управління одночасними операціями для підтримки продуктивності. Для цього програми повинні:

- Використання неблокуючих алгоритмів, де це можливо.
- Використання ефективних механізмів синхронізації з обмеженим часом очікування.
- Використання оптимізації пулу потоків для передбачуваного виконання.

4. Відмовостійкість.

Якщо система пропускає термін, це є проблемою; збій критичної системи реального часу є ще більш катастрофічним. Системи реального часу потребують механізмів швидкого виявлення збоїв та відновлення. Зазвичай це включає:

- Автоматичні вимикачі для запобігання каскадним збоям.

- Резервні механізми зі зниженою, але прийнятною продуктивністю.
- Моніторинг справності з швидким виявленням збоїв.

5. Моделі узгодженості даних.

Залежно від типу даних та рішень, що отримуються на їх основі, багато систем реального часу послаблюють сувору узгодженість для підвищення продуктивності. У таких випадках зазвичай застосовують:

- Узгоджені моделі для некритичних даних.
- Стратегії вирішення конфліктів для одночасних оновлень для підтримки цілісності даних.
- Шаблони CQRS (Command Query Responsibility Segregation) для розділення операцій читання та запису.

6. Подієво-орієнтований дизайн

Асинхронні, подієво-орієнтовані архітектури часто формують основу систем реального часу. Це означає, що кодові та архітектурні компоненти системи включатимуть:

- Брокери повідомлень, такі як Kafka або RabbitMQ, для надійної та впорядкованої доставки подій [2].
- Шаблони джерел подій для змін стану, що підлягає аудиту.
- Потокова обробка для безперервного аналізу даних.

Дотримуючись цих принципів, розробники можуть створювати системи, які відповідають потребам реального часу їхніх випадків використання. Ці шість принципів складають основні вимоги під час проектування та впровадження програм і сервісів реального часу. Крім того, розуміння різних аспектів продуктивності має вирішальне значення для системи реального часу. Це буде предметом нашого наступного обговорення.

Архітектура програмного забезпечення реального часу еволюціонувала від нішевої потреби у вбудованих системах до загальноприйнятого підходу для сучасних додатків. Оскільки компанії прагнуть надавати досвід, керований даними, здатність обробляти дані та реагувати в режимі реального часу стала ключовою конкурентною перевагою.

У цій доповіді досліджувалися основи систем реального часу: їх класифікація (жорсткі, стійкі, м'які), керівні принципи та ключові показники продуктивності.

Сучасні архітектури реального часу спираються на такі технології, як платформи потокової передачі подій (Kafka), сховища даних в пам'яті, моделі реактивного програмування та хмарні шаблони. При продуманому поєднанні ці компоненти дозволяють створювати масштабовані системи, які відповідають суворим гарантіям часу. Але чудове програмне забезпечення – це не лише технологія, а й те, як воно архітектурно спроектовано.

Щоб задовольнити вимоги систем реального часу, архітекторам потрібна постійна видимість того, як створюються та працюють додатки.

Література

1. Microservices architecture and design: A complete overview - vFunction. vFunction. URL: https://vfunction.com/blog/microservices-architecture-guide/ (date of access: 01.10.2025).

2. Ланевич Т. Apache Kafka та Rabbitmq: порівняння архітектур та можливостей // Тези XIII МНПК „Актуальні задачі сучасних технологій“, 11-12 грудня 2024 року. Тернопіль: ФОП Паляниця В. А., 2024. С. 402–403.

3. Хом’як А. С. Підвищення ефективності обробки в реальному часі у службах чат-ботів через інтеграцію черги запитів для розподілення навантаження //

Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 6-7 грудня 2023 року. – Т. : ФОП Паляниця В. А., 2023. – С. 408–409.

УДК 004.89; 004.056; 681.5
О.В. Тотосько, к.т.н., доц.; М.В. Станкевич
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗРОБЛЕННЯ ТА ДОСЛІДЖЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ «РОЗУМНИЙ БУДИНОК» З БІОМЕТРИЧНОЮ ІДЕНТИФІКАЦІЄЮ КОРИСТУВАЧІВ ТА УПРАВЛІННЯМ МІКРОКЛІМАТОМ

O.V. Totosko, Ph.D., Assoc.Prof.; M.V. Stankevych

DEVELOPMENT AND RESEARCH OF AN AUTOMATED SMART HOME SYSTEM WITH BIOMETRIC USER IDENTIFICATION AND MICROCLIMATE CONTROL

У роботі наведено результати розроблення та комплексного дослідження автоматизованої системи «Розумний будинок», у якій реалізовано інтеграцію модулів біометричної ідентифікації користувачів і системи інтелектуального керування параметрами мікроклімату. Такий підхід забезпечує одночасне підвищення рівня безпеки, комфортності та енергоефективності сучасних житлових і комерційних приміщень. Метою роботи є створення апаратно-програмного комплексу, здатного до адаптивного реагування на зміни довкілля та персональних потреб користувачів на основі аналізу даних сенсорної мережі та алгоритмів автоматичного керування.

Система створена на базі програмованого логічного контролера Arduino Mega 2560, що завдяки великій кількості входів/виходів і достатньому обсягу пам’яті дозволяє обробляти дані з множини сенсорів у реальному часі. Модульна структура системи включає блок біометричної ідентифікації, сенсорний модуль мікроклімату, блок прийняття рішень, а також набір виконавчих пристроїв – електронні реле, нагрівальний елемент, вентиляційну систему та ультразвуковий зволожувач.

Для вимірювання параметрів середовища використовуються давачі: BME280 (температура, вологість, атмосферний тиск), DHT22 (температура, вологість), BMP180 (атмосферний тиск), BH1750 (освітленість), а також газові сенсори серії MQ – MQ-2 та MQ-135 – для контролю диму, горючих газів і летких органічних сполук (VOC). Упродовж 96-годинного експериментального циклу було проведено понад 250 000 вимірювань, що дозволило отримати репрезентативні дані для аналізу стабільності роботи системи.

Середні експериментальні параметри склали: температура — 22,8 °C (діапазон 21,2–24,6 °C), відносна вологість — 47 % (коливання 38–59 %), атмосферний тиск — 993 гПа (983–1005 гПа). Освітленість у денні години сягала 550–620 лк, а ввечері знижувалася до 90–140 лк. Концентрації CO_2, виміряні MQ-135, перебували в межах 420–890 ppm, що відповідає прийнятим будівельним нормам. Також було зафіксовано, що рівень VOC у приміщенні залежав від кількості людей та інтенсивності вентиляції, змінюючись у межах 10–28 ppm.

Біометрична ідентифікація здійснюється за допомогою модуля Face Recognition Sensor FR-Cam, в основі якого лежить вбудована нейронна мережа. Модуль підтримує до 150 користувацьких профілів та демонструє високу точність. У ході досліджень середня точність розпізнавання становила 97,3 %, а час ідентифікації — 0,8–1,1 с.