При реалызацыъ проекту було використані наступні патерни:

- Стратегія
- Ланцюжок обов'язків
-

Висновки

Патерни дійсно забезпечують значне покращення процесів розробки. Їх використання допомагає розробникам створювати компоненти, що взаємодіють між собою через інтерфейси (контракти), знижуючи тим самим прямі залежності. Це забезпечує більшу гнучкість у розширенні та модифікації системи. Також патерни такі як Спостерігач, ІЗ чи Декоратор сприяють слабкому зв'язуванню компонентів, що робить систему менш чутливою до змін у будь-якій окремій частині коду.

Проте це дослідження також показало, що патерни проєктування не є універсальними рішеннями.

Ключові слова: патерни проєктування, веброзробка, автоматизація.

Використані джерела:

[1] “Мова шаблонів” (“A Pattern Language”), Крістофер Александер, 1977
[2] “Design Patterns: Elements of Reusable Object-Oriented Software”, Еріх Гамма, Річардом Хелмом, Ральфом Джонсон, Джоном Вліссідесом, 1994
[3] “Занурення в патерни проєктування”, Олександр Швець, 2022
[4] “Чиста архітектура”, Роберт Мартін, 2021

## ВПОРЯДКОВАНЕ ОБРОБЛЕННЯ ПОВІДОМЛЕНЬ У СИСТЕМАХ З РОЗПОДІЛЕНОЮ АРХІТЕКТУРОЮ / ORDERED MESSAGE PROCESSING IN SYSTEMS WITH A DISTRIBUTED ARCHITECTURE

Давиденко А.М./ Davydenko A.M.
Національний університет «Києво-Могилянська академія» /
National University of "Kyiv-Mohyla Academy" (NaUKMA)
04655, м. Київ, вулиця Григорія Сковороди, 2, НаУКМА, Факультет інформатики,
andrii.davydenko@ukma.edu.ua

*This work analyzes challenges relevant for developers of distributed systems, in particular, with regard to ensuring the order of processing of events occurring in the system and its impact on the overall state of the system. It proposes a classification of distributed systems according to the desired characteristics in terms of maintaining and evaluating the current state. An example of using the tools of modern message brokers (such as RabbitMQ and Apache Kafka), which allow the processing of messages by only one consumer, is given.*

### Вступ

Розподілена система (РС) складається з комп'ютерів (вузлів), що взаємодіють через комунікаційну мережу. На відміну від централізованих систем, де оброблення даних здійснюється одним сервером, у РС завдання розподіляються між кількома вузлами, що забезпечує масштабованість і стійкість. Проте це створює виклики, зокрема паралелізм і складність у підтриманні впорядкованих комунікацій між вузлами. Питання впорядкування повідомлень (подій) залишається актуальним і широко досліджуваним.

У цій роботі розглянуто сучасні виклики в розробці РС, проаналізовано типові сценарії, де необхідне впорядкування повідомлень, та запропоновано класифікацію РС відповідно до вимог щодо збереження стану й обчислень.

### Виклики у розробленні розподілених систем

У своїй праці Лампорт запропонував часову логіку відношення "відбулося раніше", яка є основою для концепції часткового впорядкування, або логічного годинника [2]. Цей підхід став базовим для опису алгоритмів, що визначають порядок подій у розподілених системах, і розв'язання проблем синхронізації. Важливим нововведенням стала формалізація поняття причинності в розподілених системах, де впорядкування подій забезпечується за допомогою

часових міток. Підхід, що запропонував Лампорт, дозволяє присвоювати повідомленням часові мітки, що допомагають доставляти й обробляти їх у правильному порядку, що заклало основу для впорядкування повідомлень і стимулювало подальші дослідження.

Нині існує багато алгоритмів для впорядкованого доставлення повідомлень, кожен з яких має свої переваги й недоліки, що підкреслює важливість цієї теми [1]. Однак деякі аспекти залишаються відкритими: зокрема, питання про те, чи має впорядкування повідомлень здійснюватися через проміжне програмне забезпечення або безпосередньо в прикладних програмах [5]. Порядок обробки повідомлень критично впливає на правильність функціонування розподілених систем. Важливо враховувати, що проблеми можуть лишатися непомітними на етапі розробки за відсутності належного тестування або за низьких навантажень, і проявляються тільки в умовах реальної експлуатації. Тому особливу увагу слід приділяти впорядкуванню вже на етапі проєктування систем.

## Класифікація розподілених систем

Пропонуємо класифікувати розподілені системи (РС) за ознаками, що визначають вимоги до актуального стану системи.

Перший клас РС включає системи, де потрібно забезпечити коректність стану, але допускається паралельне оброблення повідомлень. У такому випадку послідовність обробки не є критично важливою, однак необхідно відстежувати порядок повідомлень для збереження їхнього змісту. Наприклад, система, що має три стани $Off, Initalizing$ та $On$ може отримати події $E_1 = Initializing$ та $E_2 = On$ з мінімальною часовою різницею. У РС може виникнути ситуація, коли подія $E_2$ буде оброблена раніше за подію $E_1$. Задля забезпечення коректного стану системи подія $E_1$ повинна бути відкинута програмною логікою під час її оброблення як неактуальна. Використовуючи лінійні темпоральні логіки [3], очікуваний стан системи можна представити як
$F(G\ On)$. Тоді порядок повідомлень можна забезпечити за допомогою логічного годинника, такого як годинник Лампорта або векторний годинник.

Другий клас РС передбачає причинно-наслідкові залежності в поточному стані системи, що вимагає строгого порядку обробки кожного повідомлення. Якщо у РС існують чіткі умови переходу між станами (наприклад, $Off \rightarrow Initalizing$ ), пропуск події $Initalizing$ , як у попередньому прикладі, може порушити роботу системи. Враховуючи ці вимоги та використовуючи лінійні темпоральні логіки, систему можна представити як $G(Off \cup (Initalizing \cup G\ On)$. Навіть за умов масштабування потрібно забезпечити оброблення повідомлень лише одним споживачем одночасно, що суттєво обмежує пропускну здатність.

Третій клас РС також має причинно-наслідкові залежності в поточному стані, але в межах окремих контекстів. Такі обмеження є незалежними для кожного контексту, що дозволяє підвищити пропускну здатність РС. Цей підхід підходить для систем, де можна виділити незалежні об’єкти, для яких паралельна обробка подій не вплине на загальну коректність стану системи.

Завдяки такій класифікації можна вибрати оптимальний технічний підхід для взаємодії між компонентами РС ще на етапі проєктування, враховуючи специфічні вимоги до впорядкування повідомлень.

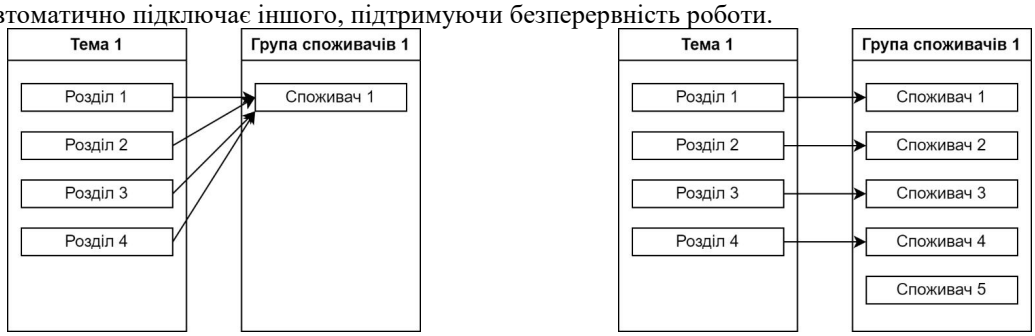
## Забезпечення впорядкованого оброблення повідомлень за допомогою брокерів повідомлень RabbitMQ та Apache Kafka

Брокери повідомлень — це проміжне програмне забезпечення, яке забезпечує обмін інформацією та взаємодію між різними системами, службами і програмами. Окрім основних функцій, брокери повідомлень можуть мати вбудовані засоби, що сприяють досягненню таких властивостей системи, як впорядковане оброблення повідомлень.

RabbitMQ — це брокер повідомлень, що реалізує асинхронний обмін між компонентами системи через протокол AMQP (Advanced Message Queuing Protocol). Виробники надсилають повідомлення до обмінників (exchanges), які спрямовують їх у відповідні черги. Черги зберігають повідомлення до моменту обробки споживачами, що дозволяє розподіляти навантаження незалежно від часу надсилання та отримання.RabbitMQ підтримує режим SAC (Single Active Consumer), де повідомлення обробляються в порядку надходження одним активним споживачем, з можливістю автоматичного перемикання на іншого у разі збою. Для підвищення пропускної здатності та забезпечення послідовності обробки в межах контексту можна використовувати плагіни з узгодженим хешуванням (consistent hashing) [4].

Apache Kafka — це розподілене сховище подій і платформа для багатопотокового оброблення з високою пропускною здатністю та низькою затримкою. На відміну від RabbitMQ, Kafka використовує модель витягування повідомлень (pull-based) та працює на основі

розподіленого журналу (log). Основні компоненти Kafka — теми (topics) і розділи (partitions). Виробник додає повідомлення до теми, а споживач самостійно вичитує їх, фіксуючи індекс останнього прочитаного повідомлення (offset). Kafka зберігає повідомлення до планового очищення, дозволяючи повторне вичитування, і не відстежує цей процес. Для впорядкованої обробки необхідно використовувати один розділ для всіх повідомлень теми та одну групу споживачів, де кожен розділ має лише одного споживача (рис. 1). У разі збою споживача Kafka автоматично підключає іншого, підтримуючи безперервність роботи.




*1* *2*

**Рис. 1.** Можливі сценарії розподілу розділів та споживачів у Apache Kafka: *1* – один споживач обробляє повідомлення з кількох розділів; *2* – один зі споживачів не є активним, інші споживачі підключені до розділів у відношенні 1:1

Підсумовуючи, сучасні брокери повідомлень надають ефективні засоби, що дозволяють забезпечити контрольований порядок оброблення повідомлень для РС другого і третього класу. Apache Kafka надає ці засоби «з коробки», у той час як для деяких сценаріїв RabbitMQ вимагає встановлення додаткових плагінів. Для підтримки РС першого класу брокери не надають вбудованих засобів. Це можна пояснити необхідністю тісної інтеграції з бізнес-логікою прикладного програмного забезпечення, адже рішення про відкидання повідомлення чи події може бути здійснено лише на рівні прикладного програмного забезпечення.

*Таблиця 1.* **Застосування RabbitMQ та Apache Kafka до класифікованих РС**

| Брокер / Клас РС | RabbitMQ | Apache Kafka |
|---|---|---|
| Клас 1 | Логічний годинник | Логічний годинник |
| Клас 2 | SAC | Єдиний розділ у темі, одна група споживачів |
| Клас 3 | SAC + плагін з узгодженим хешуванням | Окремі розділи у темі, одна група споживачів |

## Висновки

У цій роботі розглянуто актуальні виклики, з якими стикаються розробники розподілених систем та запропоновано класифікацію РС на основі вимог до збереження поточного стану. Також наведено приклади використання сучасних брокерів повідомлень, як-от RabbitMQ та Apache Kafka, для забезпечення обробки повідомлень одним споживачем, і проаналізовано можливості їх застосування до класифікованих РС.

Подальший розвиток дослідження передбачає створення дизайн-патернів для цих типів РС. Перспективним напрямом є також використання штучного інтелекту для визначення контексту стану системи і подій, які можна обробляти паралельно, не порушуючи цілісність стану.

## *Список літератури*

1. Défago X., Schiper A., Urbán P. Total order broadcast and multicast algorithms. *ACM Computing Surveys*. 2004. Vol. 36, no. 4. P. 372–421. URL: https://doi.org/10.1145/1041680.1041682 (date of access: 03.11.2024).

2. Lamport L. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*. 1978. Vol. 21, no. 7. P. 558–565. URL: https://doi.org/10.1145/359545.359563 (date of access: 03.11.2024).
3. Pnueli A. The temporal logic of programs. *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*. 1977. P. 46–57. URL: https://doi.org/10.1109/SFCS.1977.32 (date of access: 03.11.2024).
4. Vahab M., Thorup M., Zadimoghaddam M. Consistent hashing with bounded loads. *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2017. P. 587–604. URL: https://doi.org/10.1137/1.9781611975031.39 (date of access: 03.11.2024).
5. van Steen M., Tanenbaum A. S. Distributed Systems 4th edition. *DISTRIBUTED-SYSTEMS.NET*. URL: https://www.distributed-systems.net/index.php/books/ds4/ (date of access: 03.11.2024).

## ВИЯВЛЕННЯ ЗАГРОЗ «НУЛЬОВОГО ДНЯ» В УМОВАХ ОБМЕЖЕНИХ РЕСУРСІВ ВІЙСЬКОВОГО ЧАСУ /
## DETECTION OF ZERO-DAY THREATS IN A LIMITED RESOURCES OF WARTIME ENVIRONMENT

**Ісмагілов А.І.** / Ismahilov A.I.
Інститут програмних систем Національної академії наук України / Institute of Software Systems National Academy of Sciences of Ukraine
03187, Київ, просп. Академіка Глушкова, 40 корп.5, тел.: (044) 526-55-07, e-mail: ismagilov@ukr.net

The rapid growth of the number of zero-day threats, as well as their regular practical exploitation by intruders, requires timely detection of such threats, as well as further implementation of the process of automating their search, which primarily requires the development of effective and accessible approaches to assess and protect, namely in the current conditions limited resources of wartime. The work is devoted to the development of a model of safe operation of computer programs that formalize the control of processor access to memory and control over its use. The use of a modular architecture of a software package for assessing the information security of computer programs without access to their source codes, is proposed, which could provide the ability to adapt to specific conditions of software and hardware stands for testing purposes.

Аналітики безпеки Google Mandiant у 2024 році повідомили, що суб'єкти загроз демонструють кращу здатність виявляти та використовувати на практиці вразливості «нульового дня» в комп'ютерних програмах, водночас за минулий 2023 рік зі 138 вразливостей, які активно використовувалися, 97 були саме загрозами «нульового дня», що відповідно становить 70,3% від усіх загроз в цілому [1]. Оновлення безпеки від компанії Microsoft у жовтні 2024 року містило в собі виправлення для 118 вразливостей, серед яких 5 публічно розкритих та опублікованих загроз «нульового дня», дві з яких активно експлуатуються.

Доцільно відзначити, що компанія GitHub у 2024 році реалізувала рішення Copilot Autofix, яке призначено для пошуку вразливостей комп'ютерних програм саме у похідному коді на основі застосування технологій «машинного аналізу».

Наразі створення шкідливих комп'ютерних програм, які використовують вразливості «нульового дня» в комп'ютерних програмах, реалізується розробниками таких програм таким чином, щоб створені шкідливі програми не виявлялися антивірусними сканерами та моніторами, переважно такі шкідливі програми спрямовані саме на експлуатацію загроза «нульового дня».

Доцільно відзначити, що дослідження вразливостей «нульового дня» наразі є актуальним, а результати таких досліджень можуть мати суттєве значення в питаннях забезпечення національної безпеки та оборони, підвищення стійкості до кіберзагроз, пошуку нових інструментів, нових підходів та методів раннього виявлення загроз «нульового дня» в тому числі в умовах обмеженості обчислювальних ресурсів, що є притаманним в період військового часу.