

2. Angular <https://angular.dev/>
3. Vue.js - The Progressive JavaScript Framework <https://vuejs.org/>
4. Nuxt: The Intuitive Vue Framework <https://nuxt.com/>

## МЕТОДИ ЗАСТОСУВАННЯ МАШИН СТАНІВ У СИСТЕМАХ ОБРОБКИ ДАНИХ З ВИСОКИМ НАВАНТАЖЕННЯМ / METHODS FOR APPLYING STATE MACHINES IN HIGH-LOAD DATA PROCESSING SYSTEMS

Моголівський В.О. / Moholivskyi V.O.

Національний університет «Києво-Могилянська академія» / National University of Kyiv-Mohyla Academy

04070, Київ, вул. Г. Сковороди, 2, тел. (044) 425-60-59

E-mail: [v.moholivskyi@ukma.edu.ua](mailto:v.moholivskyi@ukma.edu.ua)

Methods for applying state machines to the domain of high-load data processing systems are reported in this paper. This study focuses on improving existing optimization approaches for which the use of state machines is the most beneficial. The process's in-memory cache, data processing rate limiting, and background system maintenance jobs are studied. The paper suggests utilizing state machines to alleviate the problem of distributed in-memory cache partial invalidation. Furthermore, state machines are proposed as a central control unit for managing data processing rate-limiting and long-running background tasks. The suggested methods reduce the complexity of the studied approaches by adding more transparency and monitoring capabilities.

Вимоги до систем обробки даних у сучасному світі зростають з кожним днем [3]. Обсяг та складність даних збільшується, а бажаний час їх обробки залишається незмінним [3]. Для досягнення високої ефективності у таких системах використовуються певні архітектурні підходи, техніки масштабування та методи оптимізації обробки даних [3]. Дане дослідження зосереджене на оптимізаційних підходах для яких використання машин станів є найбільш корисним. А саме на кешуванні в оперативній пам'яті процесу, контрольованому обмеженні швидкості обробки даних та фонових процесах обслуговування системи. Метою даного дослідження є доповнення кожного зі згаданих підходів використанням машин станів [2, 4] задля збільшення прозорості та ефективності їх застосування. Машини станів дозволяють керувати складністю програми роблячи процеси більш структурованими та передбачуваними.

Машина станів може бути визначена як кортеж з п'яти елементів  $(Q, \Sigma, \delta, q_0, F)$  [2], де  $Q$  – це скінченна множина станів,  $\Sigma$  – це скінченна множина вхідного алфавіту (вхідних символів або подій),  $\delta$  – це функція переходу між станами  $\delta: Q \times \Sigma \rightarrow Q$ , яка приймає вхідний символ (подію) та визначає наступний стан,  $q_0$  – це початковий стан ( $q_0 \in Q$ ),  $F$  – це множина кінцевих станів ( $F \subseteq Q$ ). У контексті розподіленої системи функція  $\delta$  окрім визначення наступного стану може також ініціювати виконання процедур згідно з функціональними вимогами програми, а саме робити виклики до баз даних, викликати віддалені процедури, надсилати повідомлення у брокери повідомлень тощо. Така машина станів може бути застосована для розв'язання функціональних задач у розподіленому середовищі.

Одним із ключових методів оптимізації обробки даних у високонавантажених системах є кешування, зокрема кешування в оперативній пам'яті [1, 3]. Основною перевагою такого підходу є швидкість доступу до кешу, вона є найвищою у порівнянні з іншими видами кешів. Проте є і суттєві недоліки такі як обмеження у розмірі та складність точкового анулювання кеша за потребою у розподілених системах. Проблема точкового анулювання кеша може бути значно полегшена за допомогою використання машини станів. Якщо відбулося оновлення даних та необхідне анулювання певної групи записів у кешах розподілених в пам'яті багатьох процесів, має бути ініціалізована машина станів, яка послідовно чи паралельно здійснить координацію видалення неактуальних записів кеша. Машина станів може використовувати виклики віддалених процедур чи обмін подіями для комунікації з кожним із процесів. Даний підхід дозволяє чітко визначити процес анулювання кеша та забезпечити структуроване керування ним у розподілених системах.

Іншим широко поширеним підходом у високонавантажених системах є контрольоване обмеження швидкості обробки даних на певних вузлах системи, включаючи як вхідні та і внутрішні вузли [3]. Машина станів може бути використана для моделювання таких обмежень. Для прикладу, розглянемо машину станів з множиною станів  $Q = \{q_0, q_1, q_2\}$ , де  $q_0$  – потік даних у рамках визначеного ліміту (обмеження не застосовуються),  $q_1$  – потік даних досяг ліміту (відбувається штучне сповільнення обробки даних),  $q_2$  – потік даних значно перевищує ліміт протягом тривалого часу (обробка даних тимчасово заблокована). Дана машина станів є спрощеним прикладом для найпростішої системи з один вузлом та базовим набором обмежень. Для практичного застосування кількість станів має бути збільшена відповідно до кількості вузлів та функціональних вимог системи. Декларативне визначення обмежень швидкості за допомогою машин станів є наочним та передбачуваним, що сильно полегшує їхнє визначення для великих розподілених систем.

Більшість високонавантажених систем з вимогою миттєвої обробки даних мають не тільки частину системи, яка працює у реальному часі, а й складову, яка містить фонові процеси обслуговування [3]. Поміж цих процесів можуть бути операції очищення чи оптимізації даних, процеси створення чи перебудови пошукових індексів, дії для збору аналітики чи звітів тощо. Типовим для таких процесів є потенційно велика тривалість на противагу основним функціям системи від яких очікують миттєве відпрацювання. Дана особливість зумовлена у першу чергу великим об'ємом даних. Але й також є закономірним наслідком оптимізації системи під певний набір функціоналу, як результат, інший функціонал на якому система не фокусувалася першочергово стає значно складнішим у реалізації. Для управління даною складністю можуть бути використані машини станів. Вони дозволяють чітко визначати кожен етап довготривалого фонових процесу. Машина станів, яка визначає процес є єдиним джерелом істини та найкращою документацією по процесу. При програмуванні довготривалих процесів без машин станів, їх обслуговуванням зазвичай займається набір функцій розкиданих по багатьох вузлах розподіленої системи. При такому підході складно швидко розуміти зв'язки між функціями без безпосереднього перерахунку їх реалізацій. Звичною практикою є написання документації для пояснення даних зв'язків. Проте документація не завжди встигає за еволюцією програмного коду. Тому використання машин станів для задач даного типу є найкращим способом дотримуватися підходу «код як документація». Також машини станів надають широкі можливості по моніторингу, візуалізації та відлагодженню фонових процесів. А також дозволяють чітко визначати набори дій для обробки помилок, які є неминучими при довготривалих процесах.

Описані методи застосування машин станів в області високонавантажених систем обробки даних вдосконалюють такі наявні підходи як кешування у пам'яті процесу, контрольоване обмеження швидкості обробки даних та виконання завдань з обслуговування системи у фоновому режимі. У роботі пропонується використання машин станів для полегшення проблеми точкового анулювання розподіленого кешу в оперативній пам'яті процесу. Також, машини станів розглядаються як центральний блок керування для контролю над обмеженнями швидкості обробки даних і довгостроковими фоновими завданнями. Запропоновані методи удосконалюють досліджувані підходи, додаючи їм більше декларативності, прозорості, можливостей моніторингу та візуалізації.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Falkevych V. G., Lisniak A. O. Methodology of cache invalidation in microservices architecture of the web applications. *Scientific notes of Taurida National V.I. Vernadsky University. Series: technical sciences*. 2023. No. 1. P. 131–135. URL: <https://doi.org/10.32782/2663-5941/2023.1/20> (date of access: 30.11.2024).

Sipser M. Introduction to the theory of computation. Michael Sipser. Thomson South-Western, 2012. 504 p.

Thatikonda V., Rajesh H. Building data-intensive applications: scalability, performance and availability. *The review of contemporary scientific and academic studies*. 2023. Vol. 3, no. 10. URL: <https://doi.org/10.55454/rcsas.3.10.2023.004> (date of access: 30.11.2024).

XState documentation. *XState.js.org*. URL: <https://xstate.js.org/docs/> (date of access: 01.11.2024).