**DOI: https://doi.org/10.15276/ict.03.2026.32**
**УДК 004.3**

# Вплив апаратного забезпечення комп'ютерної системи на вибір оптимальної архітектурної варіації розмежування відповідальності команд і запитів

**Грузін Дмитро Леонідович[1)]**
ORCID: https://orcid.org/0009-0004-8534-2559; hruzin_d@365.dnu.edu.ua. Scopus Author ID: 60077485200
**Литвинов Олександр Анатолійович[1)]**
ORCID: https://orcid.org/0000-0001-7660-1353; lytvynov_o@365.dnu.edu.ua. Scopus Author ID: 57945849600
[1)] Дніпровський національний університет імені Олеся Гончара, пр. Гагаріна, 72. Дніпро, 49000 Україна

## АНОТАЦІЯ

Наявна версія технології визначення оптимальної варіації архітектури розмежування відповідальності команд і запитів (англ. "Command Query Responsibility Segregation") у поєднанні з підходом фіксації та збереження подій (англ. ""Event Sourcing") не враховує витрати на апаратне забезпечення комп'ютерної системи, хоча вибір апаратної конфігурації впливає на продуктивність комп'ютерної системи. Метою дослідження є вдосконалити технологію шляхом доповнення процесу вибору оптимальної архітектурної варіації механізмами оцінювання апаратної складової комп'ютерної системи. Це дає змогу розглядати продуктивність як умову відповідності системи бізнес-вимогам за мінімально необхідних витрат на апаратне забезпечення. У роботі проаналізовано підходи до вибору апаратної конфігурації комп'ютерної системи. Технологію модифіковано шляхом інтегрування підходів до оцінювання апаратних конфігурацій. Модифікована технологія розділяє витрати на розроблення й супровід програмного забезпечення та витрати на придбання або оренду мінімально достатньої апаратної конфігурації та визначає оптимальну архітектурну варіацію за критерієм мінімізації сумарних витрат. Застосування модифікованої технології для двох архітектурних варіацій ("mCQRS" та "Classical CQRS") показало, що витрати на програмне забезпечення для "mCQRS" на двадцять вісім відсотків менші, ніж для "Classical CQRS", тоді як витрати на апаратну конфігурацію на тринадцять і п'ять десятих відсотка більші. Початкова версія технології не виявляє цієї відмінності, оскільки не враховує витрати на апаратне забезпечення. Запропонована модифікація підвищує обґрунтованість визначення оптимальної архітектурної варіації завдяки явному врахуванню витрат на програмну та апаратну складові комп'ютерної системи.

**Ключові слова:** комп'ютерна система; апаратне забезпечення; обчислювальні експерименти; інформаційна технологія; порівняльний аналіз; оптимізаційні моделі

***Для цитування:*** **Грузін Д. Л., Литвинов О. А."Вплив апаратного забезпечення комп'ютерної системи на вибір оптимальної архітектурної варіації розмежування відповідальності команд і запитів".** ***Інформатика. Культура. Техніка.*** **2026; Том 3 № 1 (3): 397–415. DOI: https://doi.org/10.15276/ict.03.2026.32**

**Актуальність.** У міру зростання складності сучасних комп'ютерних систем (КС) підвищуються вимоги до обґрунтованості архітектурних рішень, що ухвалюються на етапах проєктування та реалізації. Особливої актуальності ця проблема набуває для масштабованих і високопродуктивних систем, у яких вибір архітектури безпосередньо впливає не лише на функціональні можливості програмного забезпечення (ПЗ) КС, а й на вартість розроблення, трудомісткість подальшого супроводу та вимоги до кваліфікації команди розробників.

Одним із поширених підходів до побудови таких систем є архітектура розмежування відповідальності команд і запитів (англ. "Command Query Responsibility Segregation", CQRS) у поєднанні з підходом фіксації та збереження подій (англ. "Event Sourcing", ES), далі – архітектура CQRS з ES. Архітектуру CQRS з ES доцільно розглядати не як єдине універсальне рішення, а як сукупність архітектурних варіацій, які зберігають базові принципи підходу, однак відрізняються структурними рішеннями, набором застосованих шаблонів і методів, складністю розроблення й супроводу та показниками продуктивності [1]. Зазначені відмінності безпосередньо впливають на витрати, пов'язані з розробленням і супроводом системи; відповідно, необґрунтований вибір архітектурної варіації може призвести до перевищення бюджету, ускладнення супроводу системи або необхідності її подальшої архітектурної міграції.

Для зниження зазначених ризиків у роботі [1] було запропоновано технологію визначення оптимальної архітектурної варіації CQRS з ES (англ. "Decision-making Support

regarding CQRS with ES Architectural Variations", DSAV-CQRSES). Застосування цієї технології спрямоване на кількісне оцінювання архітектурних варіацій з урахуванням вимог, пріоритетів та обмежень проєкту. DSAV-CQRSES орієнтована саме на клас систем, побудованих на основі CQRS з ES, і враховує особливості цієї архітектури. Це виявляється, зокрема, у формуванні критеріїв оцінювання: частина з них ґрунтується на класах прецедентів використання, тоді як інша частина походить із проблемного простору архітектури CQRS з ES. Такий підхід забезпечує перевагу DSAV-CQRSES порівняно із загальноцільовими методами під час розв'язання задачі вибору оптимальної архітектурної варіації CQRS з ES [2], [3].

Водночас наявна версія DSAV-CQRSES має певні обмеження. Технологія дає змогу детально оцінювати програмну складову КС, однак не враховує характеристики апаратної складової обчислювального середовища. До таких характеристик належать, зокрема, кількість процесорних сокетів, кількість ядер у сокеті, кількість потоків на ядро, параметри кеш-пам'яті рівнів L1-L3, обсяг, швидкість і тип оперативної пам'яті, затримка та пропускна здатність підсистеми зберігання даних тощо. Сукупність цих параметрів безпосередньо впливає на продуктивність, яка є одним із ключових атрибутів якості високопродуктивних КС. Це підтверджується результатами експериментального порівняння хмарних обчислювальних екземплярів за критерієм співвідношення продуктивності та вартості [4]. Зокрема, показано, що вибір типу хмарного екземпляра з урахуванням характеристик навантаження може забезпечити підвищення продуктивності на 35,7 % і зниження вартості на 6,5 % порівняно з використанням одного типу екземпляра для всіх видів навантаження.

Таким чином, актуальною є задача вдосконалення технології DSAV-CQRSES у напрямі врахування не лише витрат, пов'язаних із розробленням і супроводом ПЗ, а й витрат на апаратне забезпечення (АЗ). Це дасть змогу підвищити обґрунтованість вибору рекомендованої архітектурної варіації CQRS з ES.

Наявні підходи оцінювання апаратних конфігурацій КС [5], [6], [7] доцільно умовно поділити на три групи: підходи на основі експериментального вимірювання, аналітичне моделювання продуктивності та моделювання відображення програмних компонентів на апаратні ресурси.

Перша група підходів ґрунтується на експериментальному вимірюванні продуктивності конкретної системи або її репрезентативної тестової реалізації в заданому обчислювальному середовищі. У роботі [8] наведено експериментальне оцінювання впливу архітектурних шаблонів у мікросервісних системах. Для цього автори побудували хмарну інфраструктуру, розгорнули в ній еталонну систему та реалізували кілька архітектурних шаблонів. Продуктивність оцінювалася на основі вимірювання затримки сервісів і утилізації апаратних ресурсів. У роботі [9] досліджено особливості тестування продуктивності мікросервісних застосунків на прикладі еталонного застосунку TeaStore, розгорнутого в середовищі Google Kubernetes Engine. У межах експерименту вимірювалися показники продуктивності за різних сценаріїв навантаження, зокрема 700; 800 і 900 запитів за секунду.

Водночас застосування підходів на основі експериментального вимірювання ускладнюється в разі потреби порівняння значної кількості апаратних конфігурацій. Для кожної конфігурації необхідно розгорнути відповідне обчислювальне середовище, виконати налаштування програмної системи та провести серію навантажувальних експериментів. Додатковою складністю є забезпечення коректного розподіленого навантаження, оскільки результати вимірювань можуть залежати не лише від характеристик досліджуваної конфігурації, а й від особливостей генерації запитів, кешування на різних рівнях, пропускної здатності мережі та параметрів середовища розгортання. Унаслідок цього експериментальне оцінювання великої кількості конфігурацій потребує значних часових витрат.

Друга група підходів ґрунтується на аналітичному моделюванні продуктивності КС. На відміну від підходів на основі експериментального вимірювання, такі підходи дають змогу оцінювати вплив апаратної конфігурації без обов'язкового розгортання кожного можливого варіанта обчислювального середовища. Для цього поведінка системи подається у вигляді

моделі, що враховує сценарії навантаження, інтенсивність надходження запитів, час оброблення операцій, кількість доступних ресурсів та характеристики черг.

Значну роль у розвитку цього напряму відіграли роботи з інженерії продуктивності програмного забезпечення (англ. “Software Performance Engineering”, SPE) [10], [11], [12], у яких аналіз продуктивності інтегрується в життєвий цикл розроблення систем для раннього оцінювання показників продуктивності та якості обслуговування (англ. “Quality of Service”, QoS). У межах цього напряму для оцінювання продуктивності розподілених систем широко застосовуються мережі масового обслуговування (англ. “Queueing Networks”, QN) [13] та багатошарові мережі масового обслуговування (англ. “Layered Queueing Networks”, LQN) [14]. Такі моделі дають змогу формалізувати залежність між характеристиками обчислювальних ресурсів, інтенсивністю навантаження та показниками продуктивності системи. Зокрема, у роботі [15] розглянуто моделювання та аналіз продуктивності архітектури хмарної платформи інфраструктури як послуги (англ. “Infrastructure as a Service”). Запропонована модель допомагає оцінити пропускну здатність, рівень використання ресурсів, середній час відповіді та можливі вузькі місця.

Водночас підходи на основі моделей продуктивності мають низку практичних обмежень. Для їх застосування необхідно мати достовірні вхідні параметри: інтенсивність надходження запитів, час оброблення окремих операцій, кількість і типи ресурсів, параметри черг, пропускну здатність мережі та інші характеристики обчислювального середовища. Отримання таких даних часто потребує попередніх вимірювань, калібрувальних експериментів або експертних припущень. Унаслідок цього точність оцінювання значною мірою залежить від якості вхідних параметрів і коректності припущень, покладених в основу моделі.

Третя група підходів ґрунтується на побудові повної моделі системи, у якій моделі програмних компонентів поєднуються з моделями їх розгортання та характеристиками ресурсного середовища. На відміну від аналітичних моделей продуктивності, у яких апаратна складова враховується опосередковано через параметри обслуговування (затримки, пропускну здатність тощо), такі підходи дають змогу явно описувати зв’язок між програмною системою та апаратною конфігурацією, що задається характеристиками процесорів, модулів пам’яті, мережевих та інших ресурсів.

Одним із найбільш відомих засобів такого типу є модель компонентів Palladio (англ. “Palladio Component Model”, PCM). У [16] зазначено, що Palladio-Bench дає змогу створювати моделі архітектур програмного забезпечення на основі PCM, задавати моделі розгортання та ресурсного середовища, виконувати симуляцію й отримувати оцінки продуктивності, надійності, супроводжуваності та вартості. Важливо зазначити, що розділення програмної моделі, моделі розгортання та моделі ресурсного середовища дає можливість змінювати апаратну конфігурацію або розміщення компонентів без зміни моделі програмної архітектури. Це, своєю чергою, дає змогу аналізувати вплив ресурсного середовища на показники системи.

У [17] описано PerOpteryx як засіб автоматизованого дослідження простору проєктних рішень (англ. Design Space Exploration, DSE) для PCM. У [18] розглянуто застосування PerOpteryx у промисловій розподіленій системі. Автори зазначають, що для цього було побудовано детальну модель продуктивності та вартості, яка дала змогу визначити економічно доцільне проєктне рішення. Цей результат підтверджує практичну цінність повних моделей системи для оцінювання апаратних конфігурацій, однак водночас демонструє їхню високу трудомісткість, оскільки побудова такої моделі потребує значного обсягу інформації про програмні компоненти, сценарії використання, середовище виконання та характеристики апаратних ресурсів.

Таким чином, усі три групи підходів можуть бути використані для вдосконалення DSAV-CQRSES шляхом доповнення технології механізмами оцінювання апаратної складової. Їх інтегрування доцільно розглядати як набір альтернативних механізмів, вибір між якими

залежить від умов проєкту: кількості апаратних конфігурацій, що розглядаються; наявності комплексної моделі системи; допустимої трудомісткості оцінювання.

**Об'єкт дослідження** є процес вибору архітектури високопродуктивної розподіленої КС.

**Предметом дослідження** є моделі та методи оцінювання апаратних конфігурацій КС, що застосовуються в межах DSAV-CQRSES для оптимального визначення архітектурної варіації CQRS з ES та апаратної конфігурації.

**Метою дослідження** є вдосконалення технології DSAV-CQRSES шляхом доповнення процесу визначення оптимальної архітектурної варіації CQRS з ES механізмами оцінювання апаратної складової КС. Це дасть змогу розглядати продуктивність не як показник, що підлягає максимізації, а як умову відповідності системи бізнес-вимогам за мінімально необхідних витрат на апаратне забезпечення.

Для досягнення поставленої мети необхідно розв'язати такі **задачі**:

1. Змінити метод оцінювання доцільності застосування архітектурної варіації, передбачений базовою версією DSAV-CQRSES, на метод оцінювання її інтегральної складності.

2. Додати до технології DSAV-CQRSES етап, на якому для кожної архітектурної варіації визначається мінімально достатня апаратна конфігурація, що задовольняє вимоги SLA щодо продуктивності.

3. Запропонувати підхід до мінімізації сумарних витрат на програмну та апаратну складові КС.

4. Застосувати модифіковану технологію DSAV-CQRSES на тестовому проєкті та продемонструвати вплив характеристик АЗКС на обґрунтування вибору архітектурної варіації CQRS з ES.

**Наукова новизна дослідження** полягає в удосконаленні DSAV-CQRSES шляхом інтегрування етапу розрахунку метрик продуктивності для пар «архітектурна варіація CQRS з ES – апаратна конфігурація» у процес вибору оптимальної архітектурної варіації. Розглянуті методи дають змогу оцінити продуктивність таких пар, а DSAV-CQRSES – порівняти їх з урахуванням складності ПЗ та вартості апаратної конфігурації. На відміну від базової версії технології, продуктивність використовується не як критерій максимізації, а як обмеження відповідності вимогам SLA. Це підвищує прозорість та обґрунтованість вибору архітектурної варіації CQRS з ES.

**Матеріали і методи.** Основною функцією технології DSAV-CQRSES є визначення оптимальної архітектурної варіації CQRS з ES. У наявному варіанті технології, оцінки складності реалізації та продуктивності використовуються як вхідні дані для багатокритеріального аналізу, зокрема методу аналізу ієрархій (англ. "Analytic Hierarchy Process", AHP). Такий підхід дає змогу ранжувати архітектурні варіації відповідно до пріоритетів проєкту, однак при цьому продуктивність фактично трактується як характеристика, що підлягає максимізації. Водночас для практичних задач важливо не досягти максимально можливої продуктивності, а забезпечити такий її рівень, за якого КС задовольняє вимоги угоди про рівень обслуговування (англ. "Service Level Agreement", SLA) [19]. Надлишкова продуктивність, яка не використовується бізнес-процесами, може призводити до необґрунтованого збільшення вартості апаратного забезпечення [20].

Визначальну роль у досягненні необхідного рівня продуктивності відіграє АЗКС. При цьому важливими є не лише кількісні характеристики доступних ресурсів, зокрема обсяг оперативної пам'яті та кількість процесорних ядер, а й особливості апаратної конфігурації системи: тип процесорної платформи [21] характеристики кеш-пам'яті процесора рівнів L1-L3 [22], затримка та пропускна здатність підсистеми зберігання даних [23], а також обсяг, тип, швидкодія та організація оперативної пам'яті. Ці характеристики можуть істотно впливати як на продуктивність, так і на вартість експлуатації системи, тому їх необхідно враховувати під час оцінювання архітектурних варіацій CQRS з ES.

У зв'язку з цим у роботі пропонується змінити спосіб використання показників продуктивності в межах DSAV-CQRSES. Замість безпосереднього включення продуктивності

до інтегральної оцінки архітектурної варіації як критерію, що конкурує зі складністю реалізації, продуктивність пропонується розглядати як обмеження. Для кожної архітектурної варіації необхідно визначити мінімально достатню конфігурацію АЗ, за якої система виконує вимоги SLA. Після цього вибір архітектурної варіації має здійснюватися з урахуванням двох основних груп витрат: витрат на розроблення та супровід, що визначаються складністю варіації, і витрат на експлуатацію, що визначаються необхідною апаратною конфігурацією.

Для визначення мінімально достатньої апаратної конфігурації пропонується інтегрувати в DSAV-CQRSES підходи до оцінювання апаратної складової, розглянуті в огляді літератури.

Залежно від умов проєкту до DSAV-CQRSES можуть бути інтегровані різні підходи оцінювання апаратних конфігурацій КС. Якщо множина можливих конфігурацій є обмеженою, доцільним є використання експериментального вимірювання. Наприклад, якщо система має функціонувати в межах локальної інфраструктури медичного закладу, а вибір здійснюється між кількома наявними серверними конфігураціями, немає потреби будувати моделі продуктивності. У такому випадку кількість альтернатив є обмеженою, а експериментальні вимірювання дають змогу безпосередньо перевірити виконання вимог SLA для кожної доступної конфігурації.

Для проєктів із високим рівнем формалізації, у яких уже існує детальна модель системи, доцільним є використання засобів DSE, зокрема PCM і PerOpteryx. У такому випадку витрати на побудову моделі не є додатковими для задачі оцінювання апаратної конфігурації. Прикладом є підхід безперервної інтеграції архітектурних моделей продуктивності з параметричними залежностями (англ. Continuous Integration of Architectural Performance Models with Parametric Dependencies) [24], за якого модель підтримується в актуальному стані протягом розроблення.

Якщо побудова детальної моделі системи не передбачена проєктом, а множина можливих апаратних конфігурацій є великою, доцільним є використання підходів на основі QN або LQN. Такі підходи дають змогу зменшити кількість експериментальних вимірювань, необхідних для порівняння апаратних конфігурацій.

Таким чином, запропонована модифікація DSAV-CQRSES не прив’язує технологію до одного конкретного підходу. Навпаки, вона передбачає вибір механізму оцінювання залежно від умов проєкту, кількості апаратних конфігурацій, наявності комплексної моделі системи та допустимої трудомісткості оцінювання.

Для інтегрування підходів до оцінювання апаратної конфігурації КС в DSAV-CQRSES визначено такий інтерфейс.

*Вхідні дані: множина архітектурних варіацій CQRS з ES; формалізовані класи прецедентів використання; репрезентативний тестовий проєкт (англ. Representative Test Project); множина доступних апаратних конфігурацій із відповідними вартісними характеристиками (визначається умовами конкретного проєкту, зокрема наявною локальною інфраструктурою або доступними варіантами хмарного середовища).*

*Вихідні дані: множина значень метрик продуктивності для кожної пари «архітектурна варіація – апаратна конфігурація», а саме: час відповіді сервера для команд і запитів; час від отримання сервером команди до досягнення узгодженого стану системи; частота помилок.*

У випадку застосування підходів, що потребують побудови моделі продуктивності, процес формування такої моделі може бути спрощений за рахунок використання даних, уже наявних у DSAV-CQRSES. Зокрема, технологія оперує формалізованими класами прецедентів використання, поданими за допомогою фреймової моделі знань. Кожен слот такого опису відповідає окремій активності в межах виконання прецеденту використання та пов’язаний із ресурсами КС, що залучаються під час її виконання. Це дає змогу використовувати формалізований опис прецедентів використання як основу для побудови структури моделі продуктивності.

Зокрема, ресурси, що використовуються активностями прецедентів використання класу $UC_i$, можуть бути відображені у множину станцій QN-моделі за формулами (1), (2).

$$R_{UC_i} = \bigcup S\left(\pi_{\mathcal{R}}(A_i)\right), UC_i \in UC \quad (1)$$

$$f : R_{UC_i} \to S_{UC_i}, UC_i \in UC, \quad (2)$$

де $UC$ – множина класів прецедентів використання; $UC_i$ – клас прецедентів використання; $A_i$ – множина активностей прецедентів використання класу $UC_i$; $\mathcal{R}$ – домен ресурсів КС; $\pi_{\mathcal{R}}$ – проєкція активності на ресурси, що використовуються під час її виконання; $\bigcup S\left(\pi_{\mathcal{R}}(A_i)\right)$ – об'єднання підмножин ресурсів, сформованих елементами проєкції $\pi_{\mathcal{R}}(A_i)$, у множину без повторення елементів; $f$ – функція відображення ресурсів $R_{UC_i}$ у станції QN-моделі $S_{UC_i}$; $R_{UC_i}$ – множина ресурсів, що використовуються активностями прецедентів використання класу $UC_i$; $S_{UC_i}$ – множина станцій QN-моделі для прецедентів використання класу $UC_i$.

Формули (1), (2) визначають лише множину станцій QN-моделі; маршрутизація між станціями, зокрема розгалуження та умовні переходи, задається окремо на основі структури активностей відповідного класу прецедентів використання.

*Приклад*:

КС містить такі вузли ресурсного середовища: сервер застосунку, на якому розгорнуто вебзастосунок (AppService), сховище подій (EventStore), базу даних знімків стану агрегатів (SnapshotDB) та базу даних проєкцій (ProjectionDB). Кожен із цих вузлів характеризується відповідною апаратною конфігурацією.

У Таблиці 1 наведено вузли ресурсного середовища, що залучаються під час виконання активностей класу команд створення для архітектурної варіації mCQRS [25].

*Таблиця 1.* **Відображення активностей класу команд створення на вузли ресурсного середовища для mCQRS**

| Активність | Залучені вузли ресурсного середовища |
|---|---|
| Формування команди | AppService |
| Валідація команди | AppService, SnapshotDB |
| Маршрутизація команди | AppService |
| Створення агрегату | AppService |
| Збереження агрегату | SnapshotDB, EventStore |
| Публікація команди | AppService |

***Джерело*: створено авторами**.

За формулами (1), (2) множину станцій QN-моделі для прецедентів використання з класу команд створення в архітектурній варіації mCQRS можна визначити як (3).

$$S_{CreateCmd} = \{AppService, SnapshotDB, EventStore\}. \quad (3)$$

Визначення класів навантаження QN-моделі доцільно виконувати відповідно до формалізованих класів прецедентів використання, уже наявних у DSAV-CQRSES. Це забезпечує узгодженість моделі з вхідними даними технології та дає змогу отримувати оцінки продуктивності саме для тих класів прецедентів використання, які використовуються під час порівняння архітектурних варіацій.

Після оцінювання продуктивності для пар «архітектурна варіація – апаратна конфігурація» формується множина допустимих альтернатив, серед яких вибір здійснюється за критерієм мінімізації сумарних витрат на розроблення, супровід та експлуатацію системи.

Множина альтернатив задається як (4).

$$AS = \{as_i = (v_i, M_i) \mid i \in [1; n], M_i \subseteq M\}, \quad (4)$$

де $v_i$ – архітектурна варіація CQRS з ES; $M$ – множина всіх апаратних конфігурацій КС, що розглядаються; $M_i$ – множина апаратних конфігурацій, на яких розглядається можливість розгортання системи, реалізованої на основі архітектурної варіації $v_i$; $n$ – кількість варіацій.

Для кожної альтернативи визначаються дві складові: $B_h(as_i)$ та $B_d(as_i)$, що характеризують вартість АЗ та вартість розроблення й супроводу ПЗ відповідно.

Загальна вартість реалізації альтернативи визначається як (5).

$$B(as_i) = B_h(as_i) + B_d(as_i). \quad (5)$$

Кожна апаратна конфігурація $m_{ij} \in M_i$ для архітектурної варіації $v_i$ задається парою (6).

$$m_{ij} = (c_{ij}, P_{ij}^{v_i}), \quad (6)$$

де $c_{ij}$ – вартість придбання або оренди апаратної конфігурації $m_{ij}$, а $P_{ij}^{v_i}$ – вектор характеристик продуктивності КС, програмна складова якої реалізована на основі архітектурної варіації $v_i$ та розгорнута на апаратній конфігурації $m_{ij}$.

Перевірка відповідності вимогам SLA формалізується за допомогою (7).

$$f_{SLA}: P \to \{0, 1\}, \quad (7)$$

де $f_{SLA}\left(P_{ij}^{v_i}\right) = 1$ означає, що відповідна пара «архітектурна варіація – апаратна конфігурація» задовольняє вимоги SLA, а $f_{SLA}\left(P_{ij}^{v_i}\right) = 0$ – не задовольняє.

Тоді складова вартості апаратного забезпечення $B_h(as_i)$ визначається як мінімальна вартість апаратної конфігурації, для якої відповідні характеристики продуктивності задовольняють вимоги SLA (8).

$$B_h(as_i) = \min_{m_{ij} \in M_i} \left\{ c_{ij} \mid f_{SLA}\left(P_{ij}^{v_i}\right) = 1 \right\}. \quad (8)$$

Якщо для архітектурної варіації $v_i$ не існує жодної апаратної конфігурації $m_{ij} \in M_i$, для якої виконується умова $f_{SLA}\left(P_{ij}^{v_i}\right) = 1$, відповідна альтернатива вилучається з подальшого розгляду.

У [1] під час застосування AHP було використано обернене нормування [26], оскільки відповідні показники інтерпретувалися як критерії придатності архітектурної варіації. У цій роботі така процедура не застосовується, оскільки результат AHP використовується не як інтегральна оцінка придатності, а як оцінка складності архітектурної варіації. Для подальшого розрахунку вартості розроблення та супроводу ПЗ відносна оцінка складності $rc_i$ визначається за формулою (9). Значення $rc_i$ відображає не місце варіації в загальному рейтингу, а її відносну складність щодо варіації з мінімальною складністю.

Тому за наявності більше ніж двох варіацій AHP потрібно застосовувати окремо для кожної пари «оцінювана варіація – варіація з мінімальною складністю» окремо.

$$rc_i = \frac{r_i}{r_{min}}, \quad (9)$$

де $r_i$ – оцінка складності варіації $v_i$; $r_{min}$ – оцінка складності варіації з мінімальною складністю під час одного застосування AHP; $rc_i$ – відносна складність розроблення та супроводу програмного забезпечення для варіації $v_i$.

За незмінної вартості одиниці праці витрати на розроблення та супровід ПЗ є пропорційними трудомісткості відповідних робіт [27]. Інтегральну складність архітектурної варіації можна використовувати для порівняльного оцінювання трудомісткості, оскільки вона враховує складність проєктування, зокрема використання необхідної інфраструктури, а також складність реалізації, що включає і складність тестування. Водночас у реальних проєктах зі зростанням масштабу та складності ПЗ трудомісткість його розроблення та супроводу може зростати надлінійно, що пов’язано, зокрема, зі збільшенням витрат на комунікацію та координацію між учасниками проєкту, а також інтеграцію компонентів [28].

З огляду на це в роботі прийнято припущення, що витрати на розроблення та супровід ПЗ зростають не повільніше, ніж інтегральна складність архітектурної варіації. Отже, лінійна залежність між інтегральною складністю та витратами на програмну складову відображає

мінімальний приріст додаткових витрат у разі вибору складнішої архітектурної варіації (оптимістичний прогноз).

Тоді складова вартості розроблення та супроводу ПЗ $B_d(as_i)$ визначається за формулою (10).

$$B_d(as_i) = swp * rc_i \,, \tag{10}$$

де $swp$ – базова вартість розроблення та супроводу програмного забезпечення для архітектурної варіації з мінімальною складністю.

Відповідно, задача визначення оптимальної архітектурної варіації з урахуванням апаратної складової формулюється як (11)-(13).

$$as^* = \min_{as_i \in AS} \{B_d(as_i) + B_h(as_i)\}, \tag{11}$$

за умови, що

$$\exists m_{ij} \in M_i : f_{SLA}\left(P_{ij}^{v_i}\right) = 1. \tag{12}$$

Тобто:

$$as^* = \min_{as_i \in AS} \left\{ swp * rc_i + \min_{m_{ij} \in M_i} \left[ c_{ij} \mid f_{SLA}\left(P_{ij}^{v_i}\right) = 1 \right] \right\}. \tag{13}$$

Вартості $B_d$ та $B_h$ мають бути подані в однакових одиницях вимірювання: як повна вартість або як вартість за однаковий проміжок часу.

**Результати.** Застосування модифікованої технології DSAV-CQRSES продемонстровано на прикладі порівняння двох архітектурних варіацій: mCQRS та Classical CQRS [25]. Для проведення оцінювання повторно використано формальний опис архітектурних варіацій, а також розраховані й виміряні метрики їхніх компонентів, наведені в [3].

**Експериментальне оцінювання.** Оцінювання виконано на типовому тестовому проєкті для архітектури CQRS з ES [29] для навантаження, що включає 100 запитів читання, 100 команд створення, 150 команд оновлення та 225 процесів досягнення узгодженого стану системи, спричинених виконанням команд.

Для цього навантаження задано такі вимоги SLA щодо продуктивності:

– 95-й процентиль часу відповіді сервера під час оброблення команд і запитів – не більше 15 мс;

– 95-й процентиль часу від отримання команди сервером до досягнення узгодженого стану системи – не більше 30 мс;

– частка помилок – менше 0,1 %.

Такі значення SLA обрано для наочності експерименту, оскільки вони дають змогу чітко продемонструвати вплив апаратної складової на вибір архітектурної варіації. За своїм рівнем обрані значення є характерними для реальних систем, розроблених компанією DBBSoftware.

Для зменшення впливу прикладних обчислень на результати оцінювання програмна частина системи не містить функціональних компонентів, що виконують тривалі або обчислювально складні операції. Додатково під час експерименту було вимкнено механізми програмного кешування, а вимірювання проводилися без урахування мережевої затримки. Це дало змогу зменшити вплив повторного використання даних і зовнішніх мережевих чинників на вимірювані показники продуктивності.

Для експериментального оцінювання як середовище розгортання використано хмарну платформу Amazon Web Services (AWS). З урахуванням заданих вимог SLA було сформовано множину можливих типів хмарних серверів (інстансів AWS), наведену в Таблиці 2. Кожен тип хмарного сервера було розглянуто з двома варіантами підсистеми зберігання даних (io2, gp3); у результаті сформовано шість апаратних конфігурацій для подальшого оцінювання.

Характеристики підсистем зберігання даних наведено в Таблиці 3.

*Таблиця 2*. **Характеристики типів хмарних серверів**

| Параметр | m7i.large | m7a.large | m7g.large |
|---|---|---|---|
| Архітектура | x86_64 | x86_64 | aarch64 (ARM) |
| Виробник ЦП | Intel | AMD | AWS Graviton3 |
| Модель ЦП | Xeon Platinum 8488C | EPYC 9R14 | Neoverse-V1 |
| Віртуальних ЦП (vCPU) | 2 | 2 | 2 |
| Потоків на ядро | 2 | 1 | 1 |
| Ядер на сокет | 1 | 2 | 2 |
| Сокетів | 1 | 1 | 1 |
| BogoMIPS | 4800,00 | 5199,99 | 2100,00 |
| Кеш L1d / L1i | 48 / 32 КіБ | 2 × 64 КіБ / 2 × 64 КіБ | 2 × 128 КіБ / 2 × 128 КіБ |
| Кеш L2 | 2 МіБ | 2 × 2 МіБ | 2 × 2 МіБ |
| Кеш L3 | 105 МіБ (спільний на хості) | 8 МіБ | 32 МіБ |
| Оперативна пам'ять | 8 ГБ DDR5 | 8 ГБ DDR5 | 8 ГБ DDR5 |
| Швидкість пам'яті | 4800 (gp3) / 5600 (io2) MT/c | 4800 (gp3) / 5600 (io2) MT/c | 4800 (gp3) / 5600 (io2) MT/c |
| Ціна «on-demand» для AWS регіону eu-central-1 | 0,12075 USD | 0,13886 USD | 0,0978 USD |

***Джерело*: створено авторами**

*Таблиця 3*. **Характеристики підсистем зберігання даних**

| Параметр | gp3 | io2 |
|---|---|---|
| Тип тому EBS[1] | SSD загального призначення | SSD із виділеними (provisioned) |
| IOPS[2] (в експерименті) | 3 000 | 3 000 |
| Макс. пропускна здатність (в експерименті) | 1 000 МБ/с | 1 000 МБ/с |
| Затримка (latency) | одиниці мілісекунд | Субмілісекундна[3] |
| Надійність (durability) | 99,8-99,9 % | 99,999 % |
| Розмір тому (в експерименті) | 100 ГіБ | 100 ГіБ |
| Ціна за ГБ міс для AWS регіону eu-central-1 | ≈ 0,0952 USD | ≈ 0,138 USD |
| Ціна за provisioned IOPS×міс | 3 000 IOPS безкоштовно | ≈ 0,072 USD за IOPS[4] |

***Джерело*: створено авторами**

[1] EBS – сервіс блочного сховища AWS (англ. Elastic Block Store)

[2] IOPS – операцій введення-виведення за секунду (англ. Input/Output Operations Per Second)

[3] Субмілісекундна затримка io2 Block Express забезпечується на інстансах AWS Nitro; використані інстанси m7i, m7a та m7g є Nitro-based

[4] Тариф указано для перших 32 000 provisioned IOPS; для вищих діапазонів IOPS тариф зменшується

Експеримент виконано з 3 000 IOPS для io2 дисків, що відповідає базовому рівню gp3. Для конфігураці io2 дисків метрики Amazon CloudWatch для EBS показали, що пікове значення VolumeAvgIOPS за даними CloudWatch становило 722 IOPS, а VolumeIOPSExceededCheck = 0 протягом усього експерименту. Тому для розрахунку вартості io2 було використано 1500 provisioned IOPS, що забезпечує понад двократний запас відносно виміряного пікового середнього навантаження. Оскільки субмілісекундна затримка io2 на інстансах AWS Nitro визначається типом тома та підтримуваною інфраструктурою, а не самим значенням provisioned IOPS, зміна вартості не вплинула на результати вимірювання затримки.

Пропускна здатність мережевого інтерфейсу 10 Гбіт/с приблизно відповідає 1250 МіБ/с, що перевищує максимальну пропускну здатність обох розглянутих типів томів, яка становить 1000 МіБ/с. Отже, пропускна здатність каналу доступу до EBS не є обмежувальним чинником для gp3 або io2 у використаній експериментальній конфігурації.

Для підвищення достовірності метрик продуктивності було обрано підхід експериментального вимірювання. Веб-застосунки обох архітектурних варіацій та відповідні бази даних було розгорнуто на шести апаратних конфігураціях, після чого виконано тестування продуктивності. Для кожної пари «архітектурна варіація – апаратна конфігурація» телеметричні дані зібрано в два етапи.

На першому етапі виконувалося тестування без навантаження. Підготовлений сценарій, що містив 27 команд і 13 запитів, послідовно виконувався на кожній апаратній конфігурації 100 разів.

На другому етапі проводилося тестування під навантаженням. Протягом двох хвилин на сервер щосекунди надсилалося 100 команд створення, 150 команд оновлення та 100 запитів читання. Дані, отримані для кожної конфігурації, були перевірені шляхом повторного запуску експерименту. Для конфігурації Classical CQRS m7i-gp3 було виконано п’ять повторних запусків, для всіх інших конфігурацій – по одному. Для зменшення впливу холодного старту сервера перші 10 секунд вимірювань не враховувалися. Для всіх конфігурацій, крім конфігурацій на базі m7g, значення 95-го процентиля коливалися в межах ±3 мс. Для конфігурації m7g-io2 різниця становила сотні мілісекунд, а для m7g-gp3 – секунди, що пояснюється насиченням ресурсів сервера.

Телеметричні дані, отримані під час експерименту, доступні в [30]. На їх основі розраховано такі показники: 95-й процентиль часу відповіді сервера для класів запитів читання, команд створення та команд оновлення; 95-й процентиль часу досягнення узгодженого стану системи для команд; частку помилок. Отримані метрики продуктивності під навантаженням зведено в Таблицю 4.

*Таблиця 4*. **Метрики продуктивності під навантаженням**

| Варіація | Апаратне забезпечення | Ціна (USD/місяць) | Частка помилок, % | p95 Запити | p95 Команди створення | p95 Команди оновлення | p95 Досягнення узгодженості |
|---|---|---|---|---|---|---|---|
| mCQRS | m7i-gp3 | 97,67 | 0,01 | 9,07 | 18,03 | 20,31 | 34,36 |
| | m7i-io2 | 209,95 | 0,01 | 5,54 | 12,74 | 14,58 | 23,20 |
| | **m7a-gp3** | **110,89** | **0,02** | **3,13** | **8,30** | **8,30** | **14,43** |
| | m7a-io2 | 223,17 | 0,01 | 2,25 | 5,65 | 6,02 | 10,20 |
| | m7g-gp3 | 80,91 | 2,77 | 6267,10 | 5528,47 | 6427,51 | 6442,18 |
| | m7g-io2 | 193,19 | 0,16 | 376,10 | 353,92 | 534,29 | 677,06 |
| Classical CQRS | **m7i-gp3** | **97,67** | **0,00** | **6,72** | **10,27** | **14,77** | **25,84** |
| | m7i-io2 | 209,95 | 0,01 | 5,87 | 8,23 | 12,45 | 21,79 |
| | m7a-gp3 | 110,89 | 0,00 | 2,62 | 4,84 | 6,78 | 11,79 |
| | m7a-io2 | 223,17 | 0,00 | 2,44 | 4,26 | 5,79 | 10,16 |
| | m7g-gp3 | 80,91 | 2,48 | 5381,69 | 7896,22 | 9517,94 | 8423,07 |
| | m7g-io2 | 193,19 | 0,35 | 672,70 | 662,22 | 1716,79 | 1625,26 |

***Джерело*** **створено авторами**

За результатами проведеного експерименту встановлено, що вартість мінімально достатньої апаратної конфігурації для mCQRS становить $B_h(as_{mCQRS}) = 110{,}89$ USD/місяць,

що на 13,5 % більше за відповідне значення для Classical CQRS: $B_h(as_{Classical}) = 97{,}67$ USD/місяць.

Результати експерименту показують, що конфігурація m7a забезпечує найкращі показники продуктивності для обох типів дискової підсистеми. Це пояснюється вищою однопотоковою продуктивністю процесора, зокрема найбільшим серед розглянутих конфігурацій значенням BogoMIPS, а також наявністю двох фізичних ядер без одночасної багатопотоковості. За такого підходу паралельні процеси оброблення запитів, команд і подій не конкурують за виконавчі блоки одного фізичного ядра та кеш-пам'ять рівнів L1-L2.

Конфігурація m7i демонструє проміжні результати. Її обмеження пов'язані з наявністю одного фізичного ядра з двома апаратними потоками, через що паралельні операції спільно використовують виконавчий конвеєр, регістри та кеші L1-L2 одного ядра. Великий обсяг кеш-пам'яті L3 не компенсує це обмеження, оскільки для досліджуваного сценарію робочий набір даних є відносно невеликим, а основний вплив мають саме ресурси ядра та затримки операцій введення-виведення.

Найгірші результати отримано для m7g-gp3. Це пов'язано з нижчою однопотоковою продуктивністю процесора (порівняно з m7a та m7i) та вищими затримками дискових операцій (порівняно з io2). Перехід на io2 та пришвидшення оперативної пам'яті суттєво зменшує затримку введення-виведення, тому показники m7g значно покращуються. Водночас навіть з io2 конфігурація m7g залишається повільнішою за m7a, що свідчить про обмеження з боку обчислювальної складової.

Окремо слід зазначити можливий вплив швидкості оперативної пам'яті: у конфігураціях з io2 використовувалася DDR5-5600, тоді як у конфігураціях з gp3 – DDR5-4800. Різниця у швидкості пам'яті могла частково вплинути на перевагу io2, тому цей ефект не слід повністю відносити лише до дискової підсистеми.

**Прогнозування продуктивності апаратних конфігурацій.** Для демонстрації можливості прогнозування продуктивності апаратних конфігурацій засобами моделювання QN було проведено додатковий експеримент. На основі метрик, отриманих для конфігурацій m7i-gp3, m7i-io2 та m7a-gp3 для архітектурної варіації mCQRS, було спрогнозовано характеристики продуктивності конфігурації m7a-io2.

На основі формального опису архітектурних варіацій і класів прецедентів використання, наведених у [3], за допомогою формул відображення (1)-(2) було побудовано структуру QN-моделі (Рис. 1). Моделювання виконано в середовищі Java Modeling Tools (JMT) v. 1.3.0, модуль SIMGraph. Структура моделі та класи навантаження були однаковими для всіх 4-х розглянутих альтернатив.

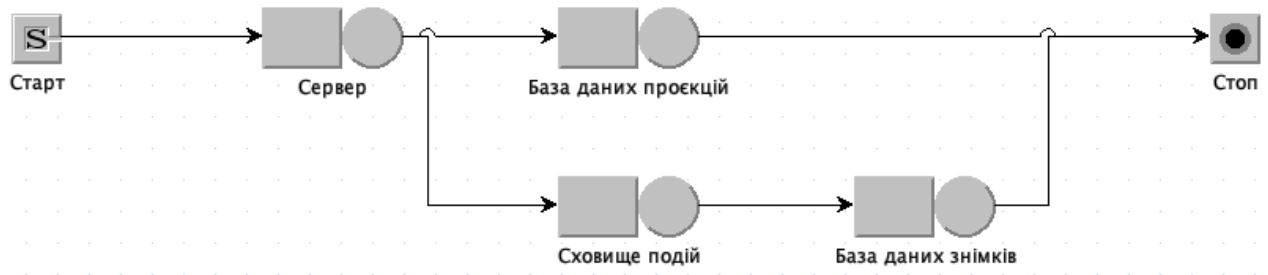


*Рис. 1.* **QN-модель архітектурної варіації mCQRS**
***Джерело*: створено авторами**

Метод прогнозування ґрунтується на масштабуванні параметрів обслуговування ресурсів під час перенесення QN-моделі на іншу апаратну конфігурацію. Подібний підхід запропоновано в роботі [31], де запропоновано масштабування "processing rate" ресурсу пропорційно відношенню benchmark-оцінок початкового та цільового ресурсу. Оскільки у QN-моделі, побудованій у JMT, параметри ресурсів задаються не через "processing rate", а

через “service demand”, було використано обернену форму цього співвідношення, причому сам коефіцієнт переходу визначався емпірично – на основі вимірювань для контрольної пари апаратних конфігурацій, а не з benchmark-оцінок.

Коефіцієнт переходу обчислювався як відношення медіан виміряного часу обслуговування на контрольній парі конфігурацій (14).

$$K_{gp3,io2} = \frac{S_{m7i,io2}}{S_{m7i,gp3}}. \quad (14)$$

Прогнозне значення «service demand» для цільової конфігурації обчислюється шляхом масштабування відповідного значення базової конфігурації (15).

$$S_{m7a,io2} = S_{m7a,gp3} K_{gp3,io2}. \quad (15)$$

Для коректного застосування K-прогнозування мають виконуватися дві умови:

– *По-перше*, для відповідних станцій і класів навантаження в усіх QN-моделях має використовуватися однаковий тип розподілу часу обслуговування. Це зумовлено тим, що різні розподіли затримок на станціях JMT [32] мають різні набори параметрів, тоді як коефіцієнт K масштабує лише значення параметра обслуговування, не змінюючи тип розподілу. Отже, застосування K-коефіцієнта є коректним лише тоді, коли для відповідних станцій змінюються тільки параметри розподілу, але не його тип.

– *По-друге*, метод ґрунтується на припущенні, що вплив зміни апаратного середовища на параметри QN-моделі є подібним для конфігурацій зі співставною обчислювальною продуктивністю. Тому відмінність між конфігураціями, за якими обчислюються коефіцієнти переходу (m7i-gp3, m7i-io2), має відповідати відмінності між базовою (m7a-gp3) та цільовою (m7a-io2) конфігураціями, для яких виконується прогнозування.

Під час вибору закону розподілу часу обслуговування на станціях QN-моделі було враховано значення квадрата коефіцієнта варіації ($CV^2$), отримані з телеметрії конфігурацій m7i-gp3, m7i-io2 та m7a-gp3. Як можливі розподіли розглядалися Erlang-k, Exponential і HyperExponential, що застосовуються в теорії масового обслуговування для апроксимації часу обслуговування за середнім значенням і коефіцієнтом варіації [33]. Зокрема, для $CV^2 < 1$ можуть застосовуватися Erlang- або mixed Erlang-розподіли, для $CV^2 \approx 1$ – Exponential-розподіл, а для $CV^2 > 1$ – HyperExponential-розподіл. Додатково розглядався LogNormal-розподіл, оскільки він показав придатність для апроксимації часу відповіді веб-систем в емпіричних дослідженнях [34].

Для підбору параметрів LogNormal-розподілу було розглянуто метод максимальної правдоподібності (англ. Maximum Likelihood Estimation, MLE) та метод моментів (англ. Method of Moments, MOM). Розподіли, побудовані за допомогою MLE і MOM, порівнювалися з емпіричними розподілами за допомогою відстані Колмогорова-Смирнова [35]. Порівняння виконано для 72 випадків: дві апаратні конфігурації (m7i-gp3, m7i-io2) × два профілі навантаження × 18 метрик у кожному профілі. Метод MLE показав кращу відповідність емпіричним даним у 67 із 72 випадків, тому його було обрано для калібрування параметрів LogNormal-розподілу в подальшому моделюванні.

Однак застосування підходу вибору розподілу, запропонованого в [33], суперечить умові K-прогнозування щодо однакового типу розподілу для відповідних станцій на різних апаратних конфігураціях. Наприклад, для станції ProjectionDB класу запитів значення $CV^2$ становить 0,89 для m7i-gp3 і 1,55 для m7a-gp3, що відповідно до підходу призводить до вибору різних законів розподілу. Тому для подальшого моделювання було обрано LogNormal-розподіл із параметрами, визначеними методом MLE. Додатково цей вибір підтверджено експериментально: для прогнозу конфігурації m7a-io2 LogNormal MLE забезпечив агреговану середню відносну похибку (англ. Mean Relative Error, MRE) 15,0% за середнім значенням і 30,0 % за 95-м процентилем, тоді як MRE для підходу [33] складає 15,7 % і 31,4 % відповідно.

Для кожної пари «станція-клас» у трьох моделях було розраховано параметри LogNormal-розподілу: параметр форми $\sigma$, що відповідає стандартному відхиленню натурального логарифма часу обслуговування, та параметр положення $\mu$, що відповідає математичному сподіванню натурального логарифма часу обслуговування. У ході серії експериментів було встановлено, що для підвищення точності моделювання ці параметри доцільно визначати на основі різних джерел телеметрії. Параметр $\sigma$ оцінювався методом MLE за метриками, отриманими під навантаженням.

Параметр $\mu$ обчислювався за формулою (16) так, щоб середнє значення LogNormal-розподілу дорівнювало медіані метрик, отриманих без навантаження (*med*).

$$\mu = ln(med) - \sigma^2/2. \quad (16)$$

Обґрунтуванням такого поділу є різна природа параметрів LogNormal-розподілу в контексті проведеного експерименту. Параметр $\mu$ визначає центр розподілу часу обслуговування і відображає вплив апаратної конфігурації. Натомість $\sigma$ характеризує варіативність часу обслуговування під навантаженням і не може бути достовірно оцінений за вимірюваннями без навантаження. Використання таких вимірювань для оцінювання $\sigma$ призводило до заниження $CV^2$ у 4-9 разів, тому $\sigma$ визначався за телеметрією, отриманою під навантаженням.

Коефіцієнти K було обчислено для всіх пар «станція-клас» за формулою (14) на основі параметрів моделей m7i-gp3 та m7i-io2. Отримані значення перебувають у діапазоні 0,625-1,033.

З використанням цих коефіцієнтів і параметрів моделі m7a-gp3 за формулою (15) було отримано прогнозовані параметри моделі для конфігурації m7a-io2, наведені в Таблиці 5.

*Таблиця 5.* **Прогнозовані значення метрик продуктивності m7a-io2**

| Показник | Час відповіді сервера для запитів | Час відповіді сервера для команд створення | Час відповіді сервера для команд оновлення | Час досягнення узгодженого стану |
|---|---|---|---|---|
| Медіана | | | | |
| Виміряне значення, мс | 0,73 | 2,29 | 2,53 | 4,49 |
| Прогнозоване значення, мс | 0,68 | 2,01 | 2,28 | 3,55 |
| MRE, % | -6,85 | -12,23 | -9,88 | -20,94 |
| 95-й процентиль | | | | |
| Виміряне значення, мс | 2,25 | 5,65 | 6,02 | 10,20 |
| Прогнозоване значення, мс | 2,19 | 5,16 | 5,48 | 7,05 |
| MRE, % | -2,67 | -8,67 | -8,97 | -30,88 |

***Джерело*: створено авторами**

* - час відповіді сервера не включає затримку мережі

Для чотирьох побудованих моделей метрики продуктивності, отримані засобами JMT, було порівняно з експериментальними вимірюваннями та розраховано середні значення MRE. Результати наведено в Таблиці 6.

*Таблиця 6.* **Середні значення MRE для метрик продуктивності**

| Апаратна конфігурація | Медіана, % | 95-й процентиль, % |
|---|---|---|
| m7i-gp3 | 4,15 | 21,53 |
| m7i-io2 | 2,04 | 11,52 |
| m7a-gp3 | 15,11 | 19,31 |
| m7a-io2 | 12,48 | 12,80 |

***Джерело*: створено авторами**

Таким чином, метрики, спрогнозовані за допомогою QN-моделі, можуть використовуватися при виборі апаратної конфігурації для зменшення кількості експериментальних вимірювань. Водночас використання моделей не усуває необхідності проведення експериментальних вимірювань. У межах цього дослідження за допомогою моделей могли бути оцінені лише 4 з 12 пар «архітектурна варіація – апаратна конфігурація». Крім того, якщо за результатами прогнозування така конфігурація визначається як оптимальна, її доцільно додатково перевірити експериментально для підтвердження виконання вимог SLA.

**Визначення оптимальної варіації.** Вибір оптимальної варіації за допомогою початкової та модифікованої версій DSAV-CQRSES виконано для двох архітектурних варіацій (mCQRS та Classical CQRS) за співвідношення операцій читання й запису 100:250, що відповідає профілю навантаження, заданому в експерименті. Для розрахунків повторно використано програмну систему, розроблену в межах дослідження [3].

Для різних пріоритетів «простота розроблення – продуктивність» системи, що відповідає середнім значенням параметричної моделі [3] початкова версія технології рекомендує mCQRS; значення інтегральної оцінки для цієї варіації перебуває в діапазоні 51,23-55,43 %.

Відповідні результати наведено в Таблиці 7.

*Таблиця 7.* **Результати застосування початкової версії DSAV-CQRSES**

| Простота розроблення / продуктивність | mCQRS | Classical CQRS |
|---|---|---|
| 90 / 10 | 55,43 | 44,57 |
| 80 / 20 | 54,38 | 45,62 |
| 50 / 50 | 51,23 | 48,77 |

***Джерело*: створено авторами**

За результатами застосування AHP у межах модифікованої технології, без урахування продуктивності як критерію та без використання оберненого нормування, оцінка складності становить 56,48 для Classical CQRS і 43,52 для mCQRS. Отже, відносна складність mCQRS дорівнює $rc_{mCQRS} = 1$ а відносна складність Classical CQRS становить $rc_{Classical} = 1{,}3$.

За результатами проведеного експерименту вартість мінімально достатньої апаратної конфігурації становить $B_h(as_{mCQRS}) = 110{,}89$ USD/місяць для mCQRS і $B_h(as_{Classical}) = 97{,}67$ USD/місяць для Classical CQRS.

Тоді задача вибору оптимальної архітектурної варіації, відповідно до (13), набуває вигляду (17).

$$as^* = \min\{swp + 110{,}89;\ 1{,}3swp + 97.67\}, \tag{17}$$

де $swp$ – базова вартість розроблення та супроводу програмного забезпечення для архітектурної варіації mCQRS (USD/місяць).

**Обговорення результатів.** Отримані результати показують, що вибір архітектурної варіації з меншою складністю програмної реалізації може призвести до зростання загальної вартості КС, якщо для виконання вимог SLA така варіація потребує дорожчої конфігурації АЗ. Водночас вплив апаратної складової залежить від конкретних вимог до продуктивності. Як видно з Таблиці 4, за інших значень SLA мінімально достатньою для обох архітектурних варіацій може бути одна й та сама апаратна конфігурація. Крім того, різниця у витратах на програмне забезпечення залежить від кількості класів прецедентів використання та кількості прецедентів використання у межах цих класів. Це підтверджує необхідність спільного врахування програмної та апаратної складових під час вибору оптимальної архітектурної варіації CQRS з ES.

У початковій версії DSAV-CQRSES продуктивність ураховується в межах багатокритеріальної оцінки. Як показано в таблиці 7, для заданого профілю навантаження початкова технологія рекомендує mCQRS: її інтегральна оцінка перебуває в діапазоні 51,23-55,43 %, тоді як для Classical CQRS – у діапазоні 44,85-48,93 %. Однак у такій постановці вплив конфігурації апаратного забезпечення на продуктивність не відображається явно, а

отже, не дає змоги окремо оцінити внесок програмної та апаратної складових у загальну вартість системи.

У модифікованій технології продуктивність розглядається не як критерій максимізації, а як обмеження SLA. Це дає змогу перейти від інтегрального ранжування до оцінювання витрат за формулами (11)-(13), у яких окремо виділено програмну та апаратну складові. За результатами AHP відносна складність Classical CQRS на 28 % більша за складність mCQRS. Водночас експериментальне оцінювання показало, що мінімально достатня апаратна конфігурація для Classical CQRS коштує 97,67 USD/місяць, тоді як для mCQRS – 110,89 USD/місяць, тобто на 13,5 % більше. Отже, модифікована технологія явно відображає компроміс між меншою складністю програмної реалізації та вищими витратами на апаратне забезпечення.

Прозоріша структура оцінювання є основною перевагою запропонованої модифікації порівняно з [1], [3]. Вона дає змогу явно визначити, за рахунок чого певна варіація є оптимальною: меншої складності програмної реалізації або меншої вартості апаратної конфігурації. Це запобігає ситуації, коли архітектурна варіація з меншою складністю програмної реалізації обирається без урахування того, що необхідна для неї апаратна конфігурація може збільшити загальну вартість КС.

Результати також підтверджують можливість використання QN-моделей як допоміжного механізму оцінювання апаратних конфігурацій. Як показано в Таблиці 5 та Таблиці 6, прогнозування продуктивності для конфігурації m7a-io2 за формулами (14)-(16) забезпечило середню відносну похибку в межах 13 %, що є прийнятним для вибору апаратної конфігурації. Це дає змогу зменшити кількість реальних вимірювань продуктивності і зосередити їх на конфігураціях, які за результатами моделювання є найперспективнішими. Водночас конфігурацію, яка за прогнозованими метриками визначається як оптимальна, доцільно додатково перевіряти експериментально для підтвердження виконання вимог SLA.

Окремою перевагою є спрощення побудови QN-моделі, оскільки частина її структури формується на основі даних, уже наявних у DSAV-CQRSES. Зокрема, структура QN-моделі, наведена на Рис. 1, була побудована на основі формалізованих класів прецедентів використання за формулами (1)-(2).

Обмеженням модифікованої технології є те, що результат її застосування відображає лише оптимізацію витрат, пов’язаних саме з вибором архітектурної варіації CQRS з ES та мінімально достатньої апаратної конфігурації для її розгортання. Витрати на розроблення інфраструктури, підсистем моніторингу, автоматизації розгортання та інших допоміжних компонентів не враховуються.

Також важливо зазначити, що в модифікованій технології, на відміну від початкової версії, терміновість розроблення КС не впливає явно на результат. Це можна розглядати як одне з її обмежень. У [1], [3] користувач міг задавати пріоритет між простотою розроблення та продуктивністю системи. Водночас складність архітектурної варіації впливає на результат через складову вартості розроблення та супроводу. Отже, якщо певна варіація має суттєво вищу складність, це збільшує її сумарну вартість за формулою (13) і зменшує ймовірність її вибору як оптимальної.

**Висновки.** Результати аналізу підходів до оцінювання апаратних конфігурацій КС показали, що як механізми оцінювання апаратної складової в технологію DSAV-CQRSES можуть бути інтегровані щонайменше три групи підходів, а саме: експериментальне вимірювання, моделювання на основі QN та підходи на основі комплексних моделей системи (наприклад PCM). Їх використання дає змогу визначати мінімально достатню апаратну конфігурацію для кожної архітектурної варіації CQRS з ES залежно від умов проєкту, кількості доступних конфігурацій і допустимої трудомісткості оцінювання.

Запропонована модифікація DSAV-CQRSES змінює роль продуктивності в процесі вибору архітектурної варіації. На відміну від початкової версії технології [1], продуктивність розглядається не як абстрактний критерій, що має бути максимізований, а як характеристика відповідності КС вимогам SLA за мінімально достатніх витрат на АЗ.

На прикладі порівняння 12 пар конфігурацій, утворених двома архітектурними варіаціями (mCQRS та Classical CQRS) і шістьма апаратними конфігураціями AWS для

типового тестового проєкту для даної архітектури, показано, що такий підхід дає змогу явно розділити витрати на розроблення та супровід ПЗКС і витрати на придбання або оренду АЗ. Це дає підстави стверджувати про підвищення прозорості та обґрунтованості визначення оптимальної архітектурної варіації CQRS з ES порівняно з базовою версією технології.

**Конфлікт інтересів:** Автори декларують, що не мають конфлікту інтересів, зокрема фінансового, особистого, авторського чи будь-якого іншого характеру, який міг би вплинути на дослідження, а також на результати, опубліковані в цій статті

**Фінансування:** Дослідження проводилося без фінансової підтримки

**Доступність даних:**

- Код програмного забезпечення, використаного під час експерименту, подано у відкритому репозиторії Zenodo: «dmytrohruzin/DSAV-CQRSES-ATAM-parametic-model: TODOs for MVP». DOI: https://doi.org/10.5281/zenodo.19039305
- Конфігурацію Terraform для розгортання апаратного та програмного забезпечення експерименту подано у відкритому репозиторії Zenodo: «Terraform configuration for “The impact of computer system hardware on the selection of an optimal CQRS with Event Sourcing Architectural Variation” experiment». DOI: https://doi.org/10.5281/zenodo.21854079
- Код програмних скриптів для запуску експерименту та аналізу журналів тестування подано у відкритому репозиторії Zenodo: «Scripts for running the experiment and logs analysis. Experiment for “The impact of computer system hardware on the selection of an optimal CQRS with Event Sourcing Architectural Variation” study». DOI: https://doi.org/10.5281/zenodo.21854199
- Телеметричні дані проведеного експерименту подано у відкритому репозиторії Zenodo: «The Impact of Computer System Hardware on the Selection of an Optimal CQRS with Event Sourcing Architectural Variation». DOI: https://doi.org/10.5281/zenodo.20329313

**Використання засобів штучного інтелекту:** Під час підготовки рукопису використовувався інструмент Chat GPT 5.5 для граматичної та стилістичної перевірки тексту. Як допоміжний засіб для написання програмних скриптів, призначених для запуску експерименту та аналізу журналів тестування використовувався інструмент Claude Code Opus 4.8. Усі фрагменти, опрацьовані за їх допомогою, були перевірені й відредаговані автором. ШІ не використовувався для генерації тексту, формул і таблиць, формування наукових положень, отримання результатів або формулювання висновків. Наукові положення, результати та висновки є власним інтелектуальним внеском авторів

## СПИСОК ЛІТЕРАТУРИ

1. Lytvynov, O. A. & Hruzin, D. L. “Decision-making on Command Query Responsibility Segregation with Event Sourcing architectural variations”. *Technology Audit and Production Reserves*, 2025; 4 (2(84)): 37–59, https://www.scopus.com/pages/publications/105014530633. DOI: https://doi.org/10.15587/2706-5448.2025.337168.

2. Hruzin, D. L. & Lytvynov, O. A. “Comparison of software architecture evaluation methods applicability in the context of CQRS with Event Sourcing architectural variations”. *Radio Electronics, Computer Science, Control*, 2026; 1: 103–120. DOI: https://doi.org/10.15588/1607-3274-2026-1-10.

3. Hruzin, D. L. & Lytvynov, O. A. “Devising a methodology to compare methods for selecting the optimal architectural variation of command query responsibility segregation with event sourcing”. *Eastern-European Journal of Enterprise Technologies*. 2026; 3 (2 (141)): 24–44, https://www.scopus.com/pages/publications/105044669244. DOI: https://doi.org/10.15587/1729-4061.2026.360407.

4. Maas, W., et al. “Investigating cloud instances to achieve optimal trade-offs between performance-cost efficiency”. *Computing*, 2025; 107 (82), https://www.scopus.com/pages/publications/85219617068. DOI: https://doi.org/10.1007/s00607-025-01444-9.

5. Ardagna, D., et al. “Quality-of-service in cloud computing: modeling techniques and their applications”. *Journal of Internet Services and Applications*, 2014; 5 (1), https://www.scopus.com/pages/publications/84908434478. DOI: https://doi.org/10.1186/s13174-014-0011-3.

6. Han, X., Yu, T. & Yan, G. “A systematic mapping study of software performance research”. *Software: Practice and Experience*, 2023; 53 (5): 1249–1270, https://www.scopus.com/pages/publications/85145400483. DOI: https://doi.org/10.1002/spe.3185.

7. Silva, F. A., et al. “Performance evaluation of cloud native applications: A systematic mapping study”. *Journal of Network and Systems Management*, 2025; 33 (4), https://www.scopus.com/pages/publications/105009973373. DOI: https://doi.org/10.1007/s10922-025-09937-w.

8. Meijer, W., Trubiani, C. & Aleti, A. “Experimental evaluation of architectural software performance design patterns in microservices”. *Journal of Systems and Software*, 2024; 218: 112183, https://www.scopus.com/pages/publications/85202513844.
DOI: https://doi.org/10.1016/j.jss.2024.112183.

9. Eismann, S., et al. “Microservices: A performance tester's dream or nightmare?” In *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE '20)*. 2020. p. 138–149, https://www.scopus.com/pages/publications/85085919253. DOI: https://doi.org/10.1145/3358960.3379124.

10. Smith, C. U. “Introduction to software performance engineering: Origins and outstanding problems”. In *M. Bernardo & J. Hillston (Eds.), Formal Methods for Performance Evaluation. SFM. Lecture Notes in Computer Science*. Berlin, Heidelberg: *Springer.* 2007; 4486, https://www.scopus.com/pages/publications/34548101755. DOI: https://doi.org/10.1007/978-3-540-72522-0_10/

11. Smith, C. U. “Software performance antipatterns in cyber-physical systems”. In Proceedings of *the ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2020. p. 173–180, https://www.scopus.com/pages/publications/85085954136. DOI: https://doi.org/10.1145/3358960.3379138.

12. Pinciroli, R., Smith, C.U. & Trubiani, C. “QN-based modeling and analysis of software performance antipatterns for cyber-physical systems”. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2021. p. 93–104, https://www.scopus.com/pages/publications/85104528326.
DOI: https://doi.org/10.1145/3427921.3450251

13. Balsamo, S. & Marin, A. “Queueing networks”. In *.M. Bernardo & J. Hillston (Eds.), Formal Methods for Performance Evaluation. SFM. Lecture Notes in Computer Science*. Berlin, Heidelberg: *Springer.* 2007; 4486, https://www.scopus.com/pages/publications/34548078780. DOI: https://doi.org/10.1007/978-3-540-72522-0_2.

14. Richart, M., et al. “LQ-GNN: A graph neural network model for response time prediction of microservice-based applications in the computing continuum”. *IEEE Transactions on Parallel and Distributed Systems.* 2025; 99: 1–12, https://www.scopus.com/pages/publications/105003588842. DOI: https://doi.org/10.1109/TPDS.2025.3564214.

15. Maiyama, K. M. & Kouvatsos, D. D. “Performance modelling and analysis of IaaS CCP architecture”. *Expert Systems with Applications*. 2026; 308: 131038, https://www.scopus.com/pages/publications/105029565256.
DOI: https://doi.org/10.1016/j.eswa.2025.131038.

16. Heinrich, R., Koziolek, A., Becker, S. & Reussner, R. “Infrastructure for modeling and analyzing the quality of software architectures”. In *Proceedings of 2-nd International Workshop on Establishing a Community-Wide Infrastructure for Architecture-Based Software Engineering (ICSE)*. Montreal Quebec, Canada. 2019; 2019: 2–5. DOI: https://doi.org/10.1109/ECASE.2019.00009.

17. Koziolek, A., Koziolek, H. & Reussner, R. H. “PerOpteryx: Automated application of tactics in multi-objective software architecture optimization”. In Proceedings of *the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*. Boulder, CO, USA. 2011. p. 33–42, https://www.scopus.com/pages/publications/79960492145.
DOI: https://doi.org/10.1145/2000259.2000267.

18. Gooijer, T., et al. “An industrial case study of performance and cost design space exploration”. In Proceedings *of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE)*. 2012. p. 205–216, https://www.scopus.com/pages/publications/84861080143. DOI: https://doi.org/10.1145/2188286.2188319.

19. Kazman, R., Klein, M. & Barbacci, M. “The architecture tradeoff analysis method”. In *Proceedings of 4th International Conference on Engineering of Complex Computer Systems (ICECCS)*. Monterey, CA, USA, 1998. p. 68–78, https://www.scopus.com/pages/publications/84985008410. DOI: https://doi.org/10.1109/ICECCS.1998.706657.

20. Funika, W., Koperek, P. & Kitowski, J. “Automated cloud resources provisioning with the use of the proximal policy optimization”. *The Journal of Supercomputing*. 2023; 79: 6674–6704, https://www.scopus.com/pages/publications/85141671694. DOI: https://doi.org/10.1007/s11227-022-04924-3.

21. Noor, J., Faysal, M. B., Amin, M. S., Tabassum, B., Khan, T. R. & Rahman, T. “Kubernetes application performance benchmarking on heterogeneous CPU architecture: An experimental review”. *High-Confidence Computing*, 2025; 5 (1): 100276, https://www.scopus.com/pages/publications/85217711775. DOI: https://doi.org/10.1016/j.hcc.2024.100276.

22. Mohamed, A. M., Mubark, N. & Zagloul, S. “Performance aware shared memory hierarchy model for multicore processors”. *Scientific Reports*. 2023; 13: 7313, https://www.scopus.com/pages/publications/85158828353. DOI: https://doi.org/10.1038/s41598-023-34297-3.

23. Ghoshal, D., Canon, R. S. & Ramakrishnan, L. “I/O performance of virtualized cloud environments”. In *Proceedings of the 2nd International Workshop on Data Intensive Computing in the Clouds (DataCloud-SC'11)*. 2011. p. 71–80, https://www.scopus.com/pages/publications/84856361507. DOI: https://doi.org/10.1145/2087522.2087535.

24. Mazkatli, M., Monschein, D., Armbruster, M., et al. “Continuous integration of architectural performance models with parametric dependencies – the CIPM approach”. *Automated Software Engineering*. 2025; 32 (54), https://www.scopus.com/pages/publications/105006909956. DOI: https://doi.org/10.1007/s10515-025-00521-9.

25. Hruzin, D. L. & Lytvynov, O. A. “Engineering of software systems based on snapshot-centric CQRS with event sourcing architecture”. *KPI Science News*. 2026; 142 (1). DOI: https://doi.org/10.20535/kpisn.2026.1.350992.

26. Vafaei, N., Ribeiro, R. A. & Camarinha-Matos, L. M. “Normalization techniques for multi-criteria decision making: Analytical hierarchy process case study”. In *L.M. Camarinha-Matos, A.J. Falcão, N. Vafaei, & S. Najdi (Eds.), Technological Innovation for Cyber-Physical Systems. (DoCEIS) IFIP Advances in Information and Communication Technology.* 2016; 470, https://www.scopus.com/pages/publications/84962068253. DOI: https://doi.org/10.1007/978-3-319-31165-4_26.

27. Huang, L. G., Boehm, B., Hu, H., Ge, J. & Lü, M. “Determining how much software assurance is enough? A value-based approach”. In *Proceedings of the Seventh International Workshop on Economics-Driven Software Engineering Research*. St. Louis Missouri. 2005. p. 1–5, https://www.scopus.com/pages/publications/84858458402.
DOI: https://doi.org/10.1145/1083091.1083095.

28. Srinadhraju, S., Mishra, S. & Satapathy, S. C. “Software cost estimation using social group optimization”. *Computer Systems Science and Engineering*. 2024; 48 (6): 1641–1668. DOI: https://doi.org/10.32604/csse.2024.055612.

29. Hruzin, D. “dmytrohruzin/DSAV-CQRSES-ATAM-parametic-model: TODOs for MVP (Version 1.0.2)”. *Zenodo*, 2026. DOI: https://doi.org/10.5281/zenodo.19039305.

30. Hruzin, D. L. “Telemetry data of experiment for “The Impact of Computer System Hardware on the Selection of an Optimal CQRS with Event Sourcing Architectural Variation” study”. *Zenodo*, 2026. DOI: https://doi.org/10.5281/zenodo.20329313.

31. Brunnert, A., Wischer, K. & Krcmar, H. “Using architecture-level performance models as resource profiles for enterprise applications”. In *Proceedings of the 10th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*. 2014. p. 53–62, https://www.scopus.com/pages/publications/84904466034.
DOI: https://doi.org/10.1145/2602576.2602587.

32. Bertoli, M., Casale, G. & Serazzi, G. “JMT: Performance engineering tools for system modeling”. *ACM SIGMETRICS Performance Evaluation Review*, 2009; 36 (4): 10–15. DOI: https://doi.org/10.1145/1530873.1530877

33. Tijms, H. C. “A first course in stochastic models”. *Chichester: John Wiley & Sons*. 2003. DOI: https://doi.org/10.1002/047001363x.ch2.

34. Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., et al. “Statistical analysis of a telephone call center: A queueing-science perspective”. *Journal of the American Statistical Association*. 2005; 100 (469): 36–50, https://www.scopus.com/pages/publications/14944380366. DOI: https://doi.org/10.1198/016214504000001808.

35. Massey, F. J. “The Kolmogorov-Smirnov test for goodness of fit”. *Journal of the American Statistical Association*. 1951; 46 (253): 68–78, https://www.scopus.com/pages/publications/84941871856. DOI: https://doi.org/10.1080/01621459.1951.10500769.

Отримано 06.07.2026
Отримано після перегляду 20.08.2026
Прийнято 28.08.2026

**DOI: https://doi.org/10.15276/ict.03.2026.32**
**UDC 004.3**

# The impact of computer system hardware on the selection of an optimal command query responsibility segregation with event sourcing architectural variation

**Dmytro L. Hruzin**[1)]
ORCID: https://orcid.org/0009-0004-8534-2559; hruzin_d@365.dnu.edu.ua. Scopus Author ID: 60077485200
**Oleksandr A. Lytvynov**[1)]
ORCID: https://orcid.org/0000-0001-7660-1353; lytvynov_o@365.dnu.edu.ua. Scopus Author ID: 57945849600
[1)] Oles Honchar Dnipro National University, 72, Haharina Ave. Dnipro, Ukraine 49000

## ABSTRACT

The existing version of the technology for decision-making support regarding architectural variations of “Command Query Responsibility Segregation” (CQRS) combined with Event Sourcing does not account for the hardware costs of a computer system when determining the optimal architectural variation. At the same time, the choice of hardware configuration affects the performance of the computer system. The objective of the study is to improve the technology by extending the process of determining the optimal architectural variation with mechanisms for evaluating the hardware component of the computer system. This shifts the role of performance from a maximization criterion to a condition for ensuring Service Level Agreement compliance with the minimum necessary hardware costs. The study analyzes approaches to selecting a hardware configuration for a computer system. The technology was modified by integrating approaches to evaluating hardware configurations of computer systems. The modified technology separates the costs of software development and maintenance from the costs of purchasing or renting hardware and determines the optimal architectural variation according to the criterion of minimizing total costs. The application of the modified technology to two architectural variations, “mCQRS” and “Classical CQRS”, showed that software costs for “mCQRS” are twenty-eight percent lower than for “Classical CQRS”, whereas hardware configuration costs are thirteen point five percent higher. The original version of the technology does not reveal this difference because it does not account for hardware costs. The proposed modification improves the objectivity of determining the optimal architectural variation by explicitly accounting for the costs of both the software and hardware components of the computer system.

**Keywords:** Computer system; hardware; computational experiments; information technology; comparative analysis; optimization models

## ІНФОРМАЦІЯ ПРО АВТОРІВ

**Грузін Дмитро Леонідович** - аспірант кафедри Електронних обчислювальних машин. Дніпровський національний університет імені Олеся Гончара, просп. Науки, 72. Дніпро, Україна, 49000
***Галузь досліджень:*** інформаційні технології, комп’ютерна інженерія

**Dmytro L. Hruzin** - PhD Student, Department of Electronic Computingю Oles Honchar Dnipro National University, 72, Nauky Ave. Dnipro, Ukraine, 49000
ORCID: https://orcid.org/0009-0004-8534-2559; hruzin_d@365.dnu.edu.ua. Scopus Author ID: 60077485200
***Research field:*** information technologies, computer engineering

**Литвинов Олександр Анатолійович** - канд. техніч. наук, доцент каф. Електронних обчислювальних машин. Дніпровський національний університет імені Олеся Гончара, пр. Науки, 72. Дніпро, Україна, 49000
***Галузь досліджень:*** інформаційні технології, комп’ютерна інженерія

**Oleksandr A. Lytvynov** - PhD, Associate Professor, Department of Electronic Computing Oles Honchar Dnipro National University, 72, Nauky Ave. Dnipro, Ukraine, 49000
ORCID: https://orcid.org/0000-0001-7660-1353; lytvynov_o@365.dnu.edu.ua. Scopus Author ID: 57945849600
***Research field:*** information technologies, computer engineering