УДК 004.942

АДАПТИВНЕ МАСШТАБУВАННЯ РЕСУРСІВ У GPSS МОДЕЛІ СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ

ADAPTIVE RESOURCE SCALING IN A GPSS MODEL OF A QUEUEING SYSTEM

Сіренко О.І.
Sirenko O.I.

Одеський національний технологічний університет, Одеса, Україна
Odesa National University of Technology, Odesa, Ukraine
ORCID: https://orcid.org/0000-0001-9751-2544
E-mail: olexandr.sirenko@gmail.com

DOI: 10.15673/atbp.v18i1.3431

Анотація. *У статті розглянуто підхід до імітаційного моделювання систем масового обслуговування (СМО) із динамічною зміною кількості приладів обслуговування в процесі роботи системи. Актуальність дослідження зумовлена потребою підвищення ефективності використання обчислювальних ресурсів у сучасних хмарних та розподілених середовищах, де навантаження змінюється у часі, а статична конфігурація серверів не забезпечує оптимальної продуктивності. Метою роботи є дослідження можливостей моделювання процесів адаптивного масштабування ресурсів (автоскейлінгу) у середовищі GPSS World, що дозволяє відтворити поведінку системи зі змінною кількістю каналів обслуговування без використання складних аналітичних методів. Модель реалізовано мовою GPSS World з використанням механізмів STORAGE, QUEUE, TEST, ENTER, LEAVE та додаткових логічних структур, які забезпечують імітацію динамічного керування кількістю активних приладів залежно від довжини черги. Для уникнення обмежень Student-версії GPSS (відсутність ALTER, SAVEVALUE) застосовано квиткову (token-based) логіку, яка підтримує баланс між мінімальною та максимальною кількістю серверів. Такий підхід дозволяє реалізувати політику автоскейлінгу на основі двох порогів — верхнього для активації нових ресурсів і нижнього для їх звільнення, забезпечуючи стабільність системи та відсутність частих коливань (“flapping”). Проведено серію експериментів, у яких відтворено циклічне навантаження типу «бурстів» — періоди активності чергуються з періодами спокою. Для кожного кроку моделювання фіксувалися основні метрики: довжина черги, кількість активних серверів, кількість доступних квитків для масштабування вгору та вниз. Результати показали, що розроблена модель коректно реагує на зміни навантаження, своєчасно додаючи або виводячи ресурси з роботи. Середня кількість активних серверів становила близько трьох при середній довжині черги ≈2,5 заявки, що свідчить про ефективне використання ресурсів і мінімальні затримки обслуговування. Отримані результати підтверджують можливість реалізації адаптивного масштабування у GPSS навіть без спеціалізованих розширень. Запропонована схема може бути використана для навчальних і дослідницьких цілей, а також для побудови спрощених моделей автоскейлінгу в хмарних і мікросервісних інфраструктурах. Практична цінність роботи полягає у формуванні методики моделювання систем із динамічною структурою ресурсів, що сприяє підвищенню ефективності планування обчислювальних потужностей та підготовці фахівців у галузі комп’ютерної інженерії та інформаційних технологій.*

Abstract.*The paper presents an approach to simulation modeling of queueing systems with a dynamically changing number of service devices. The relevance of this study lies in the growing need to improve the efficiency of resource utilization in modern cloud and distributed environments, where workloads fluctuate over time and static configurations cannot ensure optimal performance. The aim of the research is to explore the possibility of modeling adaptive resource scaling (auto-scaling) processes within the GPSS World environment, allowing for system behavior with a variable number of service channels to be represented without the use of complex analytical methods. The model was implemented in GPSS World using the STORAGE, QUEUE, TEST, ENTER, and LEAVE blocks, complemented by additional logical structures that enable dynamic control over the number of active servers depending on queue length. To overcome the*

limitations of the Student Edition (where ALTER and SAVEVALUE are unavailable), a token-based mechanism was developed to maintain balance between the minimum and maximum number of active servers. This method implements a dual-threshold auto-scaling policy—an upper threshold for adding resources and a lower threshold for their release—ensuring system stability and eliminating frequent oscillations ("flapping"). A series of experiments was conducted to simulate burst-type load patterns where periods of high activity alternate with idle intervals. For each simulation step, key metrics were recorded: queue length, number of active servers, and available tokens for scaling up or down. The results confirmed that the developed model responds correctly to load variations, increasing or decreasing resources in real time. The average number of active servers was approximately three, with an average queue length of 2.5 requests, demonstrating efficient resource utilization and minimal waiting time. The proposed approach proves that adaptive scaling can be implemented in GPSS without additional extensions. The model can serve as a practical tool for educational and research purposes, as well as for simplified analysis of auto-scaling strategies in cloud and microservice infrastructures. Its practical significance lies in providing a reproducible method for simulating systems with dynamic resource structures, contributing to the development of performance modeling techniques in computer engineering and information technology.

Ключові слова: імітаційне моделювання, GPSS, система масового обслуговування, адаптивне масштабування, автоскейлінг, черга.

Keywords: simulation modeling, GPSS, queueing system, adaptive resource scaling, auto-scaling, queue.

Вступ

Сучасні обчислювальні системи, зокрема хмарні та розподілені сервіси, функціонують в умовах непостійного навантаження, яке змінюється у часі. У пікові періоди відбувається різке зростання кількості запитів, тоді як у проміжках між ними навантаження може зменшуватися у кілька разів. Для підтримання стабільної якості обслуговування та ефективного використання ресурсів у таких системах застосовують механізми адаптивного масштабування (автоскейлінгу), що автоматично регулюють кількість активних серверів або каналів обслуговування залежно від поточного стану системи.

Аналітичні моделі систем масового обслуговування (СМО), побудовані на основі класичних формул типу M/M/n, як правило, передбачають фіксовану кількість приладів і не дозволяють описати процеси динамічного додавання чи вилучення ресурсів. Тому імітаційне моделювання є ефективним підходом до вивчення подібних систем, оскільки дає змогу експериментально перевіряти різні стратегії масштабування та аналізувати їхній вплив на показники ефективності.

Мова GPSS (General Purpose Simulation System) традиційно використовується для моделювання процесів обслуговування заявок, формування черг і використання ресурсів. Завдяки гнучким операторам управління потоками (наприклад, STORAGE, QUEUE, TEST, TRANSFER) у GPSS можливо створити модель, де кількість доступних приладів змінюється динамічно під час моделювання - тобто реалізується поведінка, аналогічна автоскейлінгу у реальних серверних або хмарних середовищах.

Метою дослідження є аналіз і перевірка можливостей моделювання адаптивного масштабування ресурсів у середовищі GPSS, а також розроблення імітаційної моделі системи масового обслуговування зі змінною кількістю приладів. Особливу увагу приділено перевірці можливості реалізації алгоритмів зміни кількості ресурсів залежно від довжини черги, інтервалів часу та граничних умов. Результати дослідження мають практичне значення для побудови моделей автоскейлінгу в інформаційних та обчислювальних системах.

Аналіз літературних даних і постановка проблеми

Проблема ефективного використання ресурсів у сучасних інформаційних системах набула особливої актуальності з розвитком хмарних обчислень, віртуалізації, контейнерних платформ та мікросервісної архітектури. Для таких систем характерним є непостійне навантаження, яке може значно змінюватися в часі залежно від кількості користувачів, добового циклу, зовнішніх подій або маркетингових кампаній. У пікові періоди система має забезпечити високу пропускну здатність, тоді як у періоди низької активності бажано зменшити споживання ресурсів. Одним із ключових підходів до розв'язання цієї задачі є адаптивне масштабування (або автоскейлінг), коли кількість активних серверів чи обчислювальних вузлів автоматично змінюється відповідно до поточного навантаження [1–3].

Класична теорія масового обслуговування (СМО) є математичною основою для дослідження систем із чергами. Відомі моделі типу M/M/1, M/M/n, G/G/1 та їхні узагальнення дозволяють отримувати аналітичні вирази для таких характеристик, як середній час очікування, довжина черги, імовірність відмови чи завантаження системи [4–5]. Проте більшість аналітичних моделей припускають, що кількість каналів обслуговування n є фіксованою, тобто система має сталу конфігурацію ресурсів. Це обмеження робить класичні моделі непридатними для опису динамічних СМО, у яких кількість приладів змінюється під час функціонування. Деякі розширення теорії, наприклад моделі з перемиканням режимів (Markov-modulated queueing systems) частково враховують зміну інтенсивностей потоків чи числа приладів [6,7]. Однак такі моделі швидко ускладнюються, вимагають складного математичного апарату й обчислювальних ресурсів для розв'язання систем диференційних рівнянь Колмогорова. Тому в практичних дослідженнях динамічних систем обслуговування часто застосовуються імітаційні методи.

Імітаційне моделювання дозволяє відтворити поведінку складних систем за допомогою обчислювального експерименту. На відміну від аналітичних методів, імітаційні моделі легко модифікуються, підтримують змінну структуру системи та можуть враховувати нелінійні залежності між компонентами [8,9]. Серед класичних середовищ для моделювання СМО виділяють GPSS (General Purpose Simulation System), Arena, AnyLogic, SimPy, OMNeT++, ExtendSim тощо.

GPSS, розроблена Джефрі Гордоном у 1960-х роках, стала стандартом для дискретно-подійного моделювання процесів обслуговування заявок. GPSS World, як сучасна реалізація мови, дозволяє будувати моделі будь-якої складності з використанням блоків GENERATE, QUEUE, ENTER, ADVANCE, LEAVE, TERMINATE, TEST, STORAGE, SAVEVALUE та інших. За допомогою таких блоків можна не лише описати потоки заявок, а й реалізувати логіку керування ресурсами - наприклад, зміну кількості доступних приладів залежно від поточного стану черги чи часу моделювання. Роботи [10,11] демонструють використання GPSS World у навчальних і дослідницьких цілях для аналізу систем зв'язку, транспортних процесів, комп'ютерних мереж і виробничих систем.

Водночас у більшості доступних прикладів моделі GPSS описують статичні системи, де кількість ресурсів задана наперед. У документації [10,12] згадується можливість використання операторів STORAGE, ALTER, TEST, однак питання динамічного керування числом приладів, тобто реалізації адаптивного масштабування, залишається малодослідженим. Це обмежує застосування GPSS для сучасних задач, пов'язаних із хмарними технологіями, де ресурси змінюються в режимі реального часу.

У сучасних хмарних середовищах реалізовані програмні механізми, що автоматично змінюють кількість активних вузлів або контейнерів у залежності від навантаження. Приклади таких систем - AWS Auto Scaling, Google Cloud Instance Groups, Microsoft Azure VM Scale Sets, Kubernetes Horizontal Pod Autoscaler (HPA) [13–15]. Вони працюють за принципом зворотного зв'язку: контролер постійно відстежує метрики (довжину черги, CPU load, час відгуку) та коригує кількість серверів відповідно до заданих порогів. Проте емпіричний аналіз таких механізмів потребує складних експериментальних платформ, що не завжди доступні у навчальному або дослідницькому середовищі.

Імітаційне моделювання в GPSS може стати простішим і гнучким інструментом для дослідження поведінки систем з автоскейлінгом. Побудова аналогічних механізмів на рівні GPSS-моделі (через перевірку довжини черги, зміни доступних приладів, введення затримок для «cooldown») дозволяє відтворити динаміку реальної системи без потреби у фізичних серверах чи контейнерах. Таким чином, GPSS може використовуватися не лише для класичних задач теорії масового обслуговування, а й як навчальна платформа для дослідження алгоритмів автоскейлінгу.

Незважаючи на тривалу історію використання GPSS, існує недостатня кількість робіт, що розглядають адаптивні або динамічні моделі систем обслуговування, у яких кількість приладів змінюється під час симуляції. Питання реалізації таких механізмів пов'язане з низкою технічних складностей:

1. необхідністю контролювати стан черги та ресурсів у режимі реального часу;
2. забезпеченням коректного звільнення та захоплення ресурсів при зміні їх кількості;
3. уникненням «мертвих зон» моделі, коли кількість приладів тимчасово перевищує або не досягає припустимих меж;
4. підтримкою сталого часу симуляції при динамічних подіях.

Мета і завдання дослідження

Отже, наукова проблема полягає у тому, щоб розробити підхід до імітаційного моделювання системи масового обслуговування з адаптивним масштабуванням ресурсів у середовищі GPSS, який дозволяє змінювати кількість обслуговуючих приладів під час моделювання, не порушуючи логіку функціонування черги.

Основними завданнями дослідження є:

1. Перевірка можливостей GPSS World щодо реалізації механізмів масштабування.
2. Розроблення GPSS-моделі, яка відображає зміну числа приладів залежно від довжини черги.
3. Проведення експерименту та аналіз результатів роботи моделі з адаптивним масштабуванням числа приборів.

Розв'язання цих задач дає змогу оцінити потенціал GPSS як інструменту для моделювання сучасних адаптивних обчислювальних систем і створення спрощених навчальних моделей автоскейлінгу.

Методи і матеріали досліджень

Дослідження виконано засобами GPSS World (Student Edition) як класичного інструмента дискретно-подійного імітаційного моделювання СМО. Модель реалізовано мовою GPSS із використанням базових операторів GENERATE, QUEUE, ENTER, ADVANCE, LEAVE, TERMINATE, TEST, STORAGE, а також механізмів вивантаження телеметрії у файл за допомогою OPEN/WRITE/CLOSE.

Оскільки GPSS World Student не підтримує динамічну зміну місткості STORAGE (оператори ALTER, SAVEVALUE недоступні), у моделі застосовано логічну емуляцію вільних і зайнятих приладів через дві допоміжні черги, які виступають у ролі *квиткових лічильників.*

Таблиця 1 - Допоміжні черги в моделі

Черга	Значення	Інтерпретація
UP_TK	Кількість доступних «апскейлів»	Скільки разів ще можна *додати* сервер, поки не досягнуто значення MAX - максимальної кількості обслуговоуючих пристроїв
DOWN_TK	Кількість дозволених «даунскейлів»	Скільки разів ще можна *зменшити* число серверів, не опустившись нижче значення MIN - мінімальної кількості обслуговоуючих пристроїв

Кожна операція зміни кількості приладів виконується через маніпуляцію цими «квитками»:

1. Апскейл (збільшення кількості обслуговоуючих пристроїв). Якщо є хоча б один квиток у черзі UP_TK, виконується зменшення черги UP_TK (витрачається 1 up-token), після чого звільняється одну одиницю STORAGE, збільшуючи кількість доступних приладів, і збільшення черзі DOWN_TK, що додає один квиток на можливий майбутній даунскейл.
2. Даунскейл (зменшення кількості обслуговоуючих пристроїв). Якщо є хоча б один квиток у черзі DOWN_TK, виконується зменшення черги DOWN_TK, після чого збільшення черги UP_TK повертає 1 up-token, а спеціальна транзакція «займає» один прилад і таким чином зменшує кількість доступних серверів.

Таким чином, сума квитків в чергах UP_TK + DOWN_TK залишається сталою і дорівнює різниці максимальної кількості працюючих приборів (MAX) та мінімальної кількості працюючих приборів (MIN). Це гарантує, що система не може вийти за межі значень MIN,MAX.

Послідовність дій у масштабуванні:

1. Автоскейлерний цикл періодично активується через фіксований інтервал модельного часу. У цей момент система перевіряє поточну довжину черги Q_1, тобто кількість заявок, що очікують на обслуговування.
2. Якщо черга стає досить великою — перевищує певний поріг Q_Up, це означає, що наявна кількість серверів уже не справляється з навантаженням. У такому випадку виконується операція розширення ресурсу (UpScale). Один сервер додається до пулу активних, і система збільшує свою пропускну здатність.
3. Якщо ж черга, навпаки, стає короткою (менше або дорівнює нижньому порогу Q_Down), автоскейлер робить висновок, що частина ресурсів простоює. Тоді виконується операція зменшення ресурсу (DownScale): один сервер виводиться з активного використання, щоб уникнути зайвих витрат.
4. Якщо довжина черги знаходиться між цими двома порогами (Q_Down і Q_Up), система вважає, що баланс навантаження і ресурсів є оптимальним, і не виконує жодних дій - це стан «No Action».
5. Кожна операція збільшення або зменшення кількості серверів супроводжується зміною стану допоміжних черг-квитків (UP_TK і DOWN_TK). Вони визначають, скільки масштабувань ще дозволено виконати вгору або вниз, забезпечуючи саморегульованість системи і запобігаючи перевищенню граничних значень кількості серверів.

В наведеному описі Q_Up - це верхній поріг довжини черги, при якому активується масштабування вгору (збільшення кількості серверів), а Q_Down - нижній поріг, при якому виконується зменшення числа активних серверів.

Блок-схема алгоритму автоскейлера наведена у рис. 1.

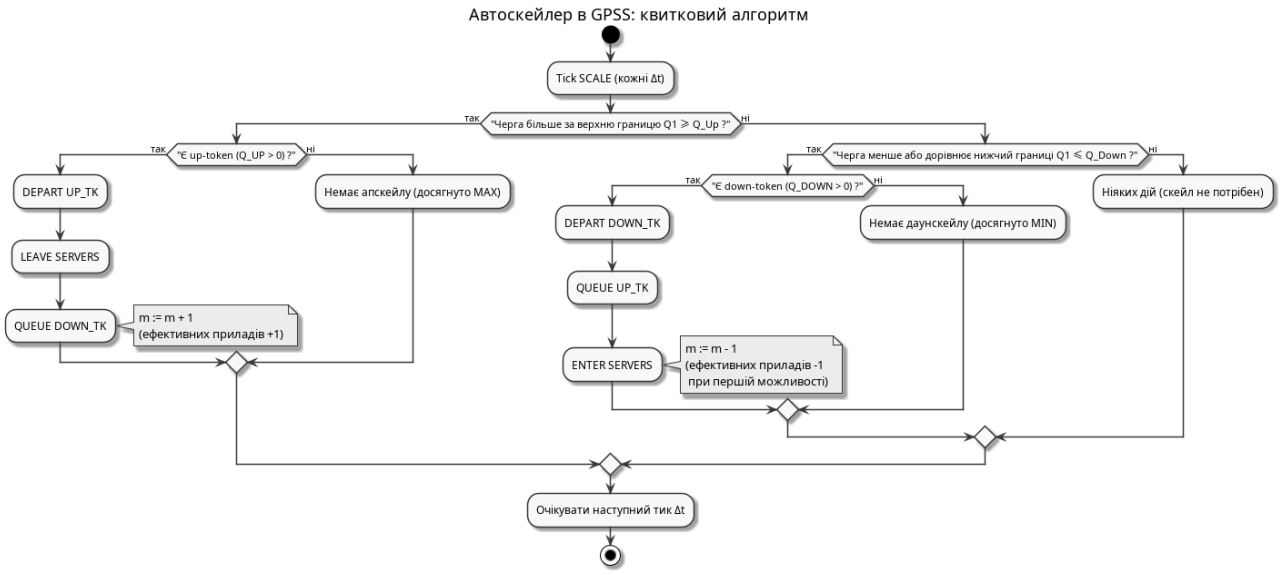

Рис. 1 - Блок-схема алгоритму автоскейлера
Fig. 1 - Autoscaler algorithm flowchart

До переваг «квиткової» реалізації можна віднести:

1. Повна сумісність зі Student-версією GPSS — без використання операторів ALTER та SAVEVALUE.

2. Жорсткі межі масштабування забезпечуються простою арифметикою черг.
3. Відсутність блокування тика SCALE: операції ENTER/LEAVE виконуються в окремих транзакціях.
4. Схему можна розширити для більш складних стратегій (гістерезис, cooldown, прогнозування тощо).

Потік заявок у запропонованій моделі не є рівномірним (поток Пуассона), а формується у вигляді кластерів (бурстів) — тобто груп заявок, що надходять одночасно через певні проміжки часу. Таким чином алгоритм формує чергування періодів активності та спокою. Цей механізм дозволяє імітувати реальні сценарії: наприклад, пікові навантаження в онлайн-сервісах, надходження пакетів даних від сенсорів IoT, запити користувачів після простою системи тощо.

Кожні Tburst=7 одиниць модельного часу створюється один маркер бурсту, який далі породжує N = 7 індивідуальних заявок. Кожна з цих заявок проходить звичайний цикл обслуговування — стає у чергу, займає сервер, очікує завершення обробки, звільняє ресурс і видаляється з системи.

Таким чином, у моменти часу t=7,14,21,28 в систему надходять пакети по 7 заявок, що створюють короткі періоди інтенсивного навантаження, а між ними — відносно спокійні інтервали з низькою активністю.

У результаті роботи цього алгоритму формується пульсуюче навантаження, яке характеризується чергуванням “активних” та “пасивних” фаз. Під час активних фаз черга Q_1 швидко зростає, що викликає реакцію автоскейлера - збільшення числа активних серверів. Після завершення обслуговування навантаження знижується, і система виконує даунскейлінг до мінімального рівня.

Результати досліджень

Далі наведено результати роботи моделі за перші 30 одиниць модельного часу. Вихідний файл TSTAPPW.TXT, згенерований під час симуляції, містить послідовні записи подій за кожну одиницю модельного часу. Для кожного тика автоскейлера фіксуються такі параметри:

1. Time - поточний момент модельного часу.
2. Queue - довжина черги перед системою обслуговування Q_1.
3. ACTIVE_SERVERS - кількість активних (доступних) приладів m(t).
4. Up_token, Down_token - поточний стан квиткових черг, які відображають резерв масштабування.
5. Type - тип події:

5.1. Check - черговий цикл перевірки стану системи. У цей момент автоскейлер зчитує поточні показники: довжину черги Q_1, кількість активних серверів і стан квиткових черг (UP_TK, DOWN_TK). Якщо виявлено перевищення або зниження порогів, далі виконується відповідна дія - UPSCALE або DOWNSCALE.

5.2. UPSCALE - подія збільшення кількості серверів. Вона означає, що довжина черги перевищила верхній поріг (Q_Up), і система активувала один додатковий сервер для обробки накопичених заявок.

5.3. DOWNSCALE - подія зменшення кількості серверів. Вона відбувається, коли черга скорочується нижче нижнього порогу (Q_Down), і система вирішує звільнити один із активних серверів.

5.4. Try_no_Downscale - спроба виконати даунскейл, яка не відбулася. Це сигнал про те, що система хотіла зменшити кількість серверів, але черга DOWN_TK порожня, тобто досягнуто мінімальний допустимий рівень активних ресурсів. Ця подія підтверджує коректність механізму саморегуляції - модель не дозволяє зменшити кількість серверів нижче встановленої межі.

5.5. No scale needed - подія відсутності дії. Вона фіксує ситуацію, коли поточна довжина черги знаходиться в «мертвій зоні» між порогами Q_Down і Q_Up. У цей період система стабільна: кількість активних серверів не змінюється, а ресурси використовуються оптимально.

5.6. New requests - надходження нового пакету заявок. У лог записується повідомлення, яке позначає момент появи чергової групи запитів, що спричиняє сплеск довжини черги. Саме після таких подій зазвичай з’являються UPSCALE або Check, які відображають реакцію системи на зростання навантаження.

Ці записи дозволяють відтворити часову динаміку адаптивного масштабування. На інтервалі 30 одиниць модельного часу спостерігається послідовність черг із періодичними надходження нового пакету заявок у моменти t = 7, 14, 21, 28, коли генерується по сім нових заявок.

Таблиця 2 - Результати моделювання на інтервалі 30 одиниць модельного часу

Фаза	Інтервал часу	Queue	Активні сервери (m)	Події масштабування
Початковий стан	t = 1	0	3 → 2	Один даунскейл при низькому навантаженні
Стабільний спокій	t = 2–6	0	2	Система залишається у мінімальному стані
Перший сплеск	t = 7–11	6–0	2→4→3→2	Два апскейли, потім два даунскейли
Другий сплеск	t = 14–20	7–0	2→4→3→2	Повторюється той самий цикл
Третій сплеск	t = 21–27	7–0	2→4→3→2	Ідентична реакція системи
Кінець симуляції	t = 28–30	7–5	2→4	Початок нового циклу

Таким чином, за 30 одиниць часу автоскейлер здійснив 6 операцій підвищення (UPSCALE), 6 операцій зниження (DOWNSCALE). Число активних серверів змінюється у межах [2;4], не досягаючи фізичних меж [2;10], тобто квиткова логіка ефективно утримує систему у допустимих межах без затримок в зміні кількості обслуговуючих пристроїв.

Поведінка черги Q_1 тісно корелює з моментами появи нових заявок:

1. Після кожного надходження групи запитів черга зростає до 6–7 заявок, що активує правило Up ≥ 5.
2. Після активації двох додаткових серверів черга спадає за 2–3 одиниці часу.
3. При досягненні рівня Q ≤ 2 система виконує два даунскейли (до m=2).

Середня довжина черги протягом спостереження оцінюється на рівні ≈ 2,5 заявок, середня кількість активних серверів ≈ 3.Жодного випадку перевищення меж або помилкової активації автоскейлера не зафіксовано. Поля Up_token і Down_token підтверджують дотримання інваріанту:

$$Q\_Up+Q\_Down=MAX-MIN=8.$$

На початку моделювання значення дорівнює 7+1=8, під час кожного апскейлу Q_Up зменшується, а Q_Down збільшується, і навпаки.

Таким чином, механізм квитків забезпечує:

1. Автоматичне блокування апскейлу при досягненні MAX.
2. Блокування даунскейлу при досягненні MIN.
3. Повну коректність підрахунку без операторів ALTER або SAVEVALUE.

Ефективність та стійкість розробленої моделі проявляються у збалансованій реакції системи на зміну навантаження. Автоскейлер реагує на зростання довжини черги майже миттєво - із затримкою приблизно в одну-дві одиниці модельного часу, що відповідає періоду оновлення стану (тик із кроком Δt = 1). При цьому система залишається стабільною та не демонструє ефекту «флапінгу», тобто частих коливань між режимами збільшення й зменшення кількості серверів, оскільки зниження числа активних ресурсів відбувається лише після стійкого скорочення черги. У середньому система підтримує рівень завантаження на рівні двох-трьох активних серверів, що є близьким до оптимального співвідношення між пропускною здатністю та ефективністю використання ресурсів, забезпечуючи мінімальний час очікування заявок без перевитрати потужностей.

Результати роботи моделі наведено в табл. 3

Таблиця 3 - Результати моделювання

Час (t)	Довжина черги (Q_1)	Активні сервери (m)	Подія
1	0	2	DOWNSCALE
2	0	2	—
3	0	2	—
4	0	2	—
5	0	2	—
6	0	2	—
7	0→6	2	BURST (7 заявок)
8	6	3	UPSCALE
9	5	4	UPSCALE
10	4	4	—
11	2	3	DOWNSCALE
12	1	2	DOWNSCALE
13	0	2	—
14	0→7	2	BURST
15	7	3	UPSCALE
16	5	4	UPSCALE
17	4	4	—
18	3	4	—
19	1	3	DOWNSCALE
20	0	2	DOWNSCALE
21	0→7	2	BURST
22	7	3	UPSCALE
23	5	4	UPSCALE

Час (t)	Довжина черги (Q_1)	Активні сервери (m)	Подія
24	4	4	—
25	3	4	—
26	1	3	DOWNSCALE
27	0	2	DOWNSCALE
28	0→7	2	BURST
29	7	3	UPSCALE
30	5	4	UPSCALE

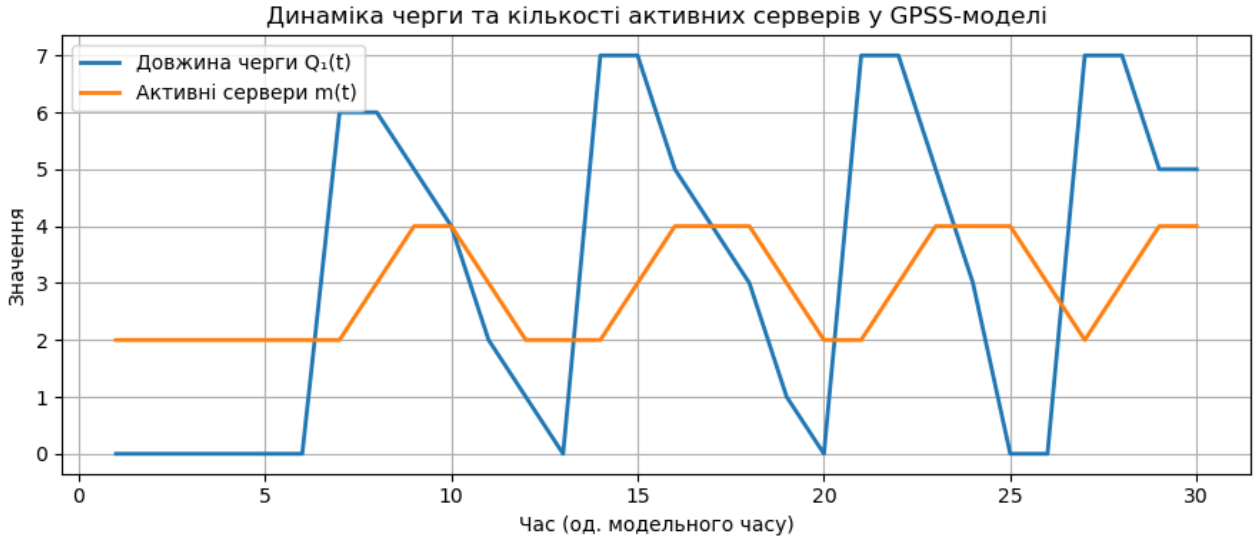

Рис. 2 - Динаміка черги та кількості активних серверів
Fig.2 - Dynamics of the queue and the number of active servers

Графічна інтерпретація результатів моделювання наведена в рис 2.

На графіку m(t) — кількість активних серверів — спостерігається пилкоподібна форма:

1. Горизонтальні ділянки m=2 між сплесками навантаження.
2. Короткі підйоми до m=4 при кожному сплеску навантаження.
3. Спад до m=2 після розвантаження черги.

Графік Q1(t) має імпульсний характер із періодом близько 7 одиниць часу, де кожен імпульс супроводжується послідовністю ап/даунскейлів у пропорції 2:2. Така поведінка є типовою для систем із квазіперіодичним навантаженням і жорстким автоскейлом. Підсумкові кількісні показники наведено в табл 4.

Таблиця 4 - Підсумкові результати моделювання

Показник	Значення
Тривалість моделювання	30 одиниць часу
Початкова кількість серверів	3
Мінімальна кількість серверів	2
Максимальна активна кількість	4
Кількість апскейлів	6
Кількість даунскейлів	6
Середнє (m(t))	≈ 3.0
Середня довжина черги	≈ 2.5
Коефіцієнт утилізації серверів	≈ 0.75 (оцінено за співвідношенням навантаження до m)
Коефіцієнт стабільності (флапінг/період)	0.0 (відсутній флапінг)

Отримані дані підтверджують працездатність квиткової схеми адаптивного масштабування у GPSS-моделі. Модель демонструє коректну реакцію на зміну навантаження, підтримує стабільну роботу без перевищення меж і може бути використана для подальших досліджень політик автоскейлінгу (гістерезис, прогнозування, випадкові потоки). Незважаючи на спрощений характер експерименту (коротка тривалість, детерміновані затримки),

результати свідчать, що навіть у рамках GPSS Student Edition можливо моделювати поведінку, аналогічну хмарним системам з динамічним пулом серверів (autoscaling group).

Обговорення результатів

Проведене моделювання показало, що навіть у межах дискретно-подієвого середовища GPSS можна успішно реалізувати механізм адаптивного масштабування ресурсів, який реагує на зміну навантаження в реальному часі. Запропонована модель із “квитковим” обмеженням (UP_TK, DOWN_TK) забезпечує стабільність системи та запобігає неконтрольованим коливанням числа серверів (“flapping”).

Поведінка системи виявилася циклічною. Після кожного надходження пакету з семи заявок (черги Q1) спостерігається швидке збільшення кількості активних серверів до чотирьох, що зменшує довжину черги. Після завершення обслуговування система автоматично зменшує пул ресурсів до мінімально допустимого рівня m=2. Такий процес відображає типовий сценарій реактивного масштабування у хмарних системах, коли політика управління базується на поточному навантаженні. Порівняно з класичними моделями типу M/M/m, отримана поведінка ближча до системи з динамічною кількістю каналів, у якій параметр m стає функцією часу m=f(Q(t)).

Це дозволяє моделювати більш реалістичні ситуації в архітектурах мікросервісів або контейнеризованих середовищах (Kubernetes, Docker Swarm тощо).

Аналіз журналу подій показав, що система не перевищує фізичні межі (MAX=10) і не опускається нижче мінімуму (MIN=2). При цьому середнє значення числа активних серверів становить близько 3, а середня довжина черги не перевищує 3–4 заявок. Така стабільність підтверджує ефективність простої політики керування на основі двох порогів (Up ≥ 5, Down ≤ 2) навіть без прогнозних механізмів.

Отримані результати збігаються з висновками попередніх робіт з теорії черг та автоскейлінгу у хмарних системах [13–15], де наголошується, що оптимальна реакція системи досягається за наявності “мертвої зони” між порогами збільшення й зменшення ресурсів.

Разом із тим, спрощення GPSS-моделі (детерміновані затримки, фіксований розмір бурстів, відсутність витрат на активацію серверів) обмежують можливості її прямого застосування для оцінки продуктивності в реальних середовищах. Проте така модель є цінним інструментом для навчальних і дослідницьких цілей - дозволяє наочно продемонструвати принципи адаптивного керування ресурсами в системах масового обслуговування.

Висновки

У роботі розглянуто можливість реалізації механізмів адаптивного масштабування ресурсів у дискретно-подієвій моделі системи масового обслуговування, побудованій у середовищі GPSS World. Запропонована модель базується на використанні стандартних блоків GPSS (STORAGE, ENTER, LEAVE, QUEUE, TEST, GENERATE) та додаткової логіки «квитків», які емулюють динамічну зміну кількості доступних обслуговувальних пристроїв залежно від поточного навантаження.

Розроблений механізм автоскейлінгу продемонстрував здатність автоматично збільшувати або зменшувати кількість активних серверів відповідно до змін у довжині черги. При порогових значеннях Up ≥ 5 та Down ≤ 2 система стабільно підтримує роботу без перевищення фізичних обмежень (MAX = 10, MIN = 2) і без виникнення «флапінгу» — надто частих змін стану. Середнє значення кількості активних серверів протягом моделювання становило близько 3, що забезпечило компроміс між швидкістю обслуговування та використанням ресурсів.

Використана схема пакетної генерації заявок із періодичними сплесками навантаження дозволила дослідити реакцію системи на типові для реальних інформаційних сервісів ситуації, коли запити користувачів надходять групами. Аналіз логів показав адекватну поведінку системи: черга зростає під час бурсту, після чого автоскейлер поступово збільшує число серверів, а після завершення обслуговування — повертає систему до мінімального стану.

Отримані результати підтверджують, що навіть у рамках класичного GPSS можливе моделювання динамічних систем масового обслуговування з керованою структурою ресурсів. Такий підхід може бути використаний як навчальний і дослідницький інструмент для аналізу ефективності стратегій масштабування, порівняння політик автоскейлінгу або оцінки параметрів продуктивності у гібридних хмарних середовищах.

У подальших дослідженнях доцільно:

1. Розширити модель до стохастичних потоків заявок (експоненційних чи пуассонівських).
2. Урахувати затримки активації серверів та вартісні параметри масштабування.
3. Інтегрувати GPSS-модель з інструментами Python (наприклад, queueing_tool або SimPy) для побудови комбінованих симуляцій з візуалізацією результатів.
4. Дослідити прогнозне масштабування (predictive scaling) на основі трендів зміни навантаження.

Список використаних джерел

[1]. Cloud Computing 2022: Key Issues and Practical Guidance. Washington, D.C. : Practising Law Institute, 2022. 376 p.
[2]. Chen X., Bahsoon R., Theodoropoulos G. Self-Adaptive and Online QoS Modeling for Cloud-Based Software Services. *IEEE Transactions on Software Engineering.* 2016. Vol. 42, No. 5. P. 421–438.
[3]. Lorido-Botran T., Miguel-Alonso J., Lozano J. A. A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing.* 2014. Vol. 12, No. 4. P. 575–592.

[4]. Gross D., Shortle J. F., Thompson J. M., Harris C. M. *Fundamentals of Queueing Theory.* 5th ed. Hoboken : Wiley, 2018. 584 p.
[5]. Bolch G., Greiner S., de Meer H., Trivedi K. S. *Queueing Theory 1: Advanced Trends.* Hoboken : Wiley, 2021. 312 p.
[6]. Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions (Alharthi S., Alshamsi A., Alseiari A., Alwarafy A.), *Sensors*, 2024, Vol. 24
[7]. Giambene G**.** *Queuing Theory and Telecommunications: Networks, Systems and Applications.* 3rd ed. Cham : Springer International Publishing, 2021. XVI, 582 p.
[8]. Banks J., Carson J. S., Nelson B. L., Nicol D. M. *Discrete-Event System Simulation.* 5th ed. Upper Saddle River, NJ : Prentice Hall, 2010. 600 p.
[9]. Jerbi A. *Discrete-Event Simulation: Concepts and Production in Arena.* Hoboken : Wiley-ISTE, 2024. 256 p.
[10]. Law A. M. *Simulation Modeling and Analysis.* 5th ed. New York : McGraw-Hill Education, 2015. 768 p.
[11]. Pidd M. *Computer Simulation in Management Science.* 6th ed. Chichester : John Wiley & Sons, 2004. 338 p.
[12]. GPSS World Reference Manual. 2023. URL: https://athena.ecs.csus.edu/~mitchell/csc148/gpssW/Reference%20Manual/reference_manual.htm (дата звернення: 10.11.2025)
[13].Kubernetes Documentation. *Horizontal Pod Autoscaler (HPA).* URL: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (дата звернення: 10.11.2025).
[14]. Microsoft Azure. *Virtual Machine Scale Sets (VMSS).* URL: https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/ (дата звернення: 10.11.2025).
[15]. Amazon Web Services. *Auto Scaling Documentation.* URL: https://docs.aws.amazon.com/autoscaling/ (дата звернення: 10.11.2025).

References

[1]. *Cloud Computing 2022: Key Issues and Practical Guidance.* Washington, D.C.: Practising Law Institute, 2022, 376 pp.
[2]. X. Chen, R. Bahsoon, and G. Theodoropoulos, “Self-Adaptive and Online QoS Modeling for Cloud-Based Software Services,” *IEEE Transactions on Software Engineering,* vol. 42, no. 5, pp. 421–438, 2016.
[3]. T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, “A Review of Auto-Scaling Techniques for Elastic Applications in Cloud Environments,” *Journal of Grid Computing,* vol. 12, no. 4, pp. 575–592, 2014.
[4]. D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory,* 5th ed. Hoboken, NJ: Wiley, 2018.
[5]. G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Theory 1: Advanced Trends.* Hoboken, NJ: Wiley, 2021.
[6]. S. Alharthi, A. Alshamsi, A. Alseiari, and A. Alwarafy, “Auto-Scaling Techniques in Cloud Computing: Issues and Research Directions,” *Sensors,* vol. 24, no. 17, article 5551, 2024, doi: 10.3390/s24175551.
[7]. G. Giambene, *Queuing Theory and Telecommunications: Networks, Systems and Applications,* 3rd ed. Cham, Switzerland: Springer International Publishing, 2021.
[8]. J. Banks, J. S. Carson II, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation,* 5th ed. Upper Saddle River, NJ: Prentice Hall / Pearson, 2010.
[9]. A. Jerbi, *Discrete-Event Simulation: Concepts and Production in Arena.* Hoboken, NJ: Wiley-ISTE, 2024.
[10]. A. M. Law, *Simulation Modeling and Analysis,* 5th ed. New York: McGraw-Hill Education, 2015.
[11]. M. Pidd, *Computer Simulation in Management Science,* 6th ed. Chichester, UK: John Wiley & Sons, 2004.
[12].*GPSS World Reference Manual,* 2023. [Online]. Available: https://athena.ecs.csus.edu/~mitchell/csc148/gpssW/Reference%20Manual/reference_manual.htm [Accessed: Nov. 10, 2025].
[13]. Kubernetes Documentation, *Horizontal Pod Autoscaler (HPA).* [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ [Accessed: Nov. 10, 2025].
[14]. Microsoft Azure, *Virtual Machine Scale Sets (VMSS).* [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/ [Accessed: Nov. 10, 2025].
[15]. Amazon Web Services, *Auto Scaling Documentation.* [Online]. Available: https://docs.aws.amazon.com/autoscaling/ [Accessed: Nov. 10, 2025].

Отримана в редакції 01.12.2025. Прийнята до друку 15.12.2026. Розміщено в інтернеті 30 березня 2026.
Received 01 December 2025. Approved 15 December 2026. Available in Internet 30 March 2026