

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ
СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Курсова робота з освітнього компоненту
«Моделювання систем. Курсова робота»

**Тема: Імітаційна модель системи процесорів на основі
формального опису мережею Петрі**

Керівник:

ст.викл. Дифучина Олександра Юріївна

«Допущено до захисту»

«___» _____ 2024 р.

Захищено з оцінкою

Члени комісії:

Виконавець:

Панченко Сергій Віталійович
студент групи ІП-11
залікова книжка № ІП-1123

«5» грудня 2024 р.

Інна СТЕЦЕНКО

Олександра ДИФУЧИНА

Київ – 2024

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна «Моделювання систем»

Спеціальність 121 Інженерія програмного забезпечення

Курс 4 Група ІП-11 Семестр 1

ЗАВДАННЯ

на курсову роботу студента

Панченка Сергія Віталійовича
(прізвище, ім'я, по батькові)

1. Тема роботи «Імітаційна модель системи процесорів на основі формального опису мережею Петрі»

2. Термін здачі студентом закінченої роботи "5" грудня 2024р.

3. Зміст розрахунково-пояснювальної записки

1. Опис 2. Псевдокод 3. Реалізація 4. Реалізація 5. Реалізація 6. Проведення експериментів над моделями. Висновки.

4. Дата видачі завдання "29" жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання курсової роботи	Термін виконання етапів роботи	Примітка
1	Отримання індивідуального завдання на курсову роботу	29.10.2024	
2	Розробка концептуальної моделі системи	29.11.2024	
3	Розробка формалізованої моделі системи	03.11.2024	
4	Алгоритмізація моделі системи та її програмна реалізація	08.11.2024	
5	Експериментальне дослідження моделі системи	13.11.2024	
6	Інтерпритація результатів моделювання, формулювання висновків та пропозицій	15.11.2024	
7	Оформлення пояснювальної записки	17.11.2024	
8	Захист КР	5.12.2024	

Студент _____ Панченко С. В.
(підпис)

Керівник _____ Дифучина О.Ю.
(підпис)

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка курсової роботи складається з 5 розділів, містить 42 рисунки, 13 таблиць, 2 додатки, 7 джерел.

Мета: визначення статистичних характеристик роботи багатопроцесорної обчислювальної системи, зокрема середнього значення, середнього квадратичного відхилення та закону розподілу для таких вихідних значень, як: часу виконання завдання в системі; завантаження чотирьох дисків, каналу передачі даних і процесорів; використання пам'яті; кількості завдань, які очікують виділення ресурсу, та часу їхнього очікування.

У першому розділі представлено концептуальну модель багатопроцесорної обчислювальної системи, визначено її основні компоненти та характеристики, а також взаємодію між ними.

У другому розділі створено формалізовану модель системи на основі мереж Петрі, що включає опис позицій, переходів та стохастичних параметрів.

У третьому розділі розроблено алгоритм імітації мережі Петрі, реалізовано програмну модель системи та внесено вдосконалення для підтримки аналізу специфічних задач.

У четвертому розділі проведено експериментальне дослідження моделі, включаючи аналіз часу виконання завдань, завантаження компонентів і статистичних характеристик системи.

У п'ятому розділі здійснено інтерпретацію результатів моделювання, визначено ефективність системи та запропоновано напрямки для її оптимізації.

КЛЮЧОВІ СЛОВА: МОДЕЛЮВАННЯ СИСТЕМ, МЕРЕЖІ ПЕТРІ, БАГАТОПРОЦЕСОРНА ОБЧИСЛЮВАЛЬНА СИСТЕМА.

ЗМІСТ

Постановка завдання.....	8
Вступ.....	10
1 Розробка концептуальної моделі.....	12
1.1 Опис задачі.....	12
1.2 Структурна схема моделі.....	12
1.3 Опис процесу.....	12
1.4 Вхідні змінні.....	13
1.5 Вихідні змінні.....	13
1.6 BPNM-діаграма.....	14
1.7 Висновки до розділу.....	16
2 Розробка формалізованої моделі.....	17
2.1 Вхідні параметри.....	18
2.2 Вихідні параметри.....	20
2.2.1 Часи виконання завдання в системі та очікування виділення пам'яті.....	20
2.2.1.1 Середнє значення.....	20
2.2.1.2 Середнє квадратичне відхилення.....	21
2.2.2 Кількості завдань в очікуванні пам'яті та зайнятих сторінок.....	21
2.2.2.1 Середнє значення.....	21
2.2.2.2 Середнє квадратичне відхилення.....	21
2.2.3 Завантаження процесорів, дисків та каналу передачі.....	23
2.2.3.1 Середнє значення.....	23
2.2.3.2 Середнє квадратичне відхилення.....	24
2.3 Мережа Петрі.....	24
2.4 Висновки до розділу.....	26
3 Алгоритмізація моделі та її реалізація.....	27
3.1 Опис алгоритму імітації мережі Петрі.....	27
3.2 Модифікації алгоритму.....	30

3.2.1 Додавання розподілу Пуассона для генерації подій.....	30
3.2.2 Вдосконалення роботи з NetLibrary.....	30
3.2.3 Покращення графічного інтерфейсу.....	31
3.2.4 Клас CourseWorkPetriSim.....	31
3.2.4.1 Опис атрибутів.....	32
3.2.4.2 Опис методів.....	32
3.2.5 Клас CourseWorkNet. Протокол подій.....	34
3.2.5.1 Опис атрибутів.....	34
3.2.5.2 Опис TaskObject.....	36
3.2.5.3 Опис методів.....	36
3.3 Верифікація.....	38
3.3.1 Збір даних. Клас GatherDataCourseWorkNet.....	38
3.3.2 Обробка даних.....	40
3.3.3 Аналіз даних.....	45
3.3.3.1 Середні значення.....	45
3.3.3.2 Середньоквадратичні відхилення.....	46
3.4 Висновки до розділу.....	47
4 Експериментальне дослідження моделі.....	49
4.1 Визначення перехідного періоду.....	49
4.1.1 Параметри за замовчуванням.....	49
4.1.1.1 Вихідні параметри в момент часу.....	50
4.1.2 Змінені параметри.....	60
4.2 Визначення кількості необхідних прогонів.....	67
4.3 Визначення середніх значень та середньоквадратичних відхилень.....	67
4.4 Визначенні типів розподілів.....	69
4.4.1 Аналіз розподілів навантаження диска, процесора, каналу передачі.....	72
4.4.2 Аналіз розподілу кількості завдань в очікуванні пам'яті.....	72
4.4.3 Аналіз розподілу кількості часу завдання в системі.....	72
4.4.4 Аналіз розподілу часу очікування виділення пам'яті.....	75

4.4.5 Аналіз розподілу кількості зайнятих сторінок.....	77
4.5 Висновки до розділу.....	77
5 Інтерпретація результатів моделювання та експериментів.....	78
5.1 Час виконання завдань у системі.....	78
5.2 Завантаження компонентів системи.....	78
5.3 Використання пам'яті.....	78
5.4 Кількість завдань в очікуванні пам'яті.....	79
5.5 Висновки до розділу.....	79
Висновки.....	80
Список використаних джерел.....	81
ДОДАТОК А	82
ДОДАТОК Б	108

ПОСТАНОВКА ЗАВДАННЯ

Багатопроцесорна обчислювальна система складається з двох процесорів із загальною оперативною пам'яттю обсягом 131 сторінка, чотирьох накопичувачів на дисках, кожний із яких доступний обом процесорам, і одного каналу передачі даних. Завдання надходять у систему із середньою інтенсивністю, рівною 12 завданням у хвилину відповідно до розподілу Пуассона. Загальний час, необхідний процесору на обробку завдання, розподілено нормально з математичним сподіванням 10 секунд та середнім квадратичним відхиленням 3 секунди. Час обробки процесором включає переривання, необхідні для здійснення обміну по каналу вводу-виводу. Інтервали між перериваннями розподілені за негативний експоненціальний розподілом з математичним сподіванням, що дорівнює оберненій величині середньої інтенсивності операцій вводу-виводу завдання. Середня інтенсивність операцій введення-виведення розподілена рівномірно на інтервалі від 2 секунд до 10 секунд. Операції введення-виведення призначаються конкретному диску. Завданню, що надходить у систему, призначається пріоритет, що є величиною, оберненою до потреби в пам'яті. Потреба завдання в пам'яті розподілена рівномірно в інтервалі від 20 до 60 сторінок. Як тільки пам'ять виділена для завдання, один з вільних процесорів починає його обробку. При видачі запиту на здійснення введення-виведення завдання може продовжувати використання процесора доти, доки в черзі залишиться тільки один запит. Таким чином, якщо зроблений запит на здійснення введення-виведення і один запит вже очікує в черзі, то процесор звільняється, а запит на введення-виведення розміщується в черзі. Після виконання поточного запиту введення-виведення процесор може відновити обробку завдання в тому випадку, якщо вона вільна. Після переривання процесора автоматично виконується запит введення-виведення з призначеним завданню диском. Таким чином, здійснюється прямий доступ до диска з процесора. Передбачається, що час позиціонування диска розподілено рівномірно на інтервалі від 0,0 до 0,075 секунд. Одночасно може здійснюватися

тільки одна операція позиціонування диска. Після позиціонування здійснюється обмін даними по каналу передачі даних. Час обміну дорівнює $0,001 \times (2,5 + h)$, де h - рівномірно розподілена на інтервалі від 0 до 25 величина. Після здійснення обміну запит введення-виведення вважається виконаним. Визначити загальний час виконання завдання в системі, а також статистичні оцінки завантаження усіх чотирьох дисків, каналу передачі даних та обох процесорів. Крім того, необхідно одержати оцінку середнього використання пам'яті, статистику щодо кількості завдань, які очікують виділення ресурсу, та щодо часу очікування.

ВСТУП

У сучасному світі багатопроцесорні обчислювальні системи відіграють ключову роль у вирішенні складних обчислювальних задач. Ефективність їх роботи залежить від багатьох факторів, включаючи управління пам'яттю, розподіл процесорного часу та організацію введення-виведення. Тому дослідження характеристик таких систем є актуальним завданням для оптимізації їх роботи та підвищення продуктивності.

Метою даної курсової роботи є визначення статистичних характеристик роботи багатопроцесорної обчислювальної системи, зокрема середнього значення, середнього квадратичного відхилення та закону розподілу для таких вихідних значень, як-от: часу виконання завдання в системі; завантаження чотирьох дисків, каналу передачі даних і процесорів; використання пам'яті; кількості завдань, які очікують виділення ресурсу, та часу їхнього очікування.

Для досягнення поставленої мети використовується апарат мереж Петрі, який є потужним математичним інструментом для моделювання та аналізу паралельних та розподілених систем. Мережі Петрі дозволяють ефективно описувати та досліджувати асинхронні, паралельні процеси в системі, враховуючи стохастичну природу процесів, що відбуваються в ній. У роботі застосовуються різні типи ймовірнісних розподілів для моделювання надходження завдань (розподіл Пуассона), часу обробки завдань (нормальний розподіл), інтервалів між перериваннями (експоненційний розподіл) та інших характеристик системи.

Для реалізації моделі використовується спеціалізоване програмне забезпечення PetriObjModelPaint, розроблене для моделювання та аналізу мереж Петрі. Цей інструмент надає можливості для створення, візуалізації та дослідження мережевих моделей, а також дозволяє отримувати статистичні оцінки завантаження всіх компонентів системи, включаючи процесори, диски, канал передачі даних, аналізувати використання пам'яті та характеристики черг завдань.

Результати дослідження дозволять оцінити ефективність роботи системи та виявити потенційні "вузькі місця" в її функціонуванні, що може бути використано для подальшої оптимізації роботи багатопроцесорних обчислювальних систем.

1 РОЗРОБКА КОНЦЕПТУАЛЬНОЇ МОДЕЛІ

У цьому розділі представлено концептуальну модель багатопроцесорної обчислювальної системи, яка включає опис її складових, основних процесів, правил взаємодії елементів та параметрів, необхідних для подальшого моделювання і аналізу.

1.1 Опис задачі

Мета моделювання – визначити показники роботи багатопроцесорної обчислювальної системи, зокрема:

- загальний час виконання завдань;
- завантаження дисків, каналу передачі даних і процесорів;
- середнє використання пам'яті;
- кількість завдань, що очікують на виділення ресурсу;
- час очікування виділення пам'яті.

1.2 Структурна схема моделі

Об'єктом моделювання є багатопроцесорна система, яка включає:

- два процесори, що використовують спільну оперативну пам'ять на 131 сторінку;
- чотири диски, доступні обом процесорам;
- один канал передачі даних, який використовується для обміну між процесорами та дисками.

1.3 Опис процесу

Процес складається з таких елементів:

- завдання надходять у систему з інтенсивністю 12 завдань/хв за розподілом Пуассона;
- кожному завданню призначається пріоритет, обернений до його потреби в пам'яті;

- потреба завдань у пам'яті розподілена рівномірно в межах 20–60 сторінок;
- якщо пам'ять доступна, одразу призначається один із вільних процесорів;
- час виконання завдання розподілений нормально із середнім 10 секунд та відхиленням 3 секунди;
- у процесі виконання завдань виникають переривання для обміну даними; інтервали між перериваннями розподілені експоненційно з параметром, оберненим до інтенсивності операцій введення-виведення (2–10 секунд);
- операції введення-виведення включають;
- позиціонування диска (0–0,075 с);
- передачу даних ($0,001 \times (2,5 + h)$), де h розподілено рівномірно в межах 0–25);
- завдання звільняє процесор, якщо черга операцій введення-виведення складається з максимум одного завдання; в іншому випадку, воно продовжує займати процесор.

1.4 Вхідні змінні

Модель має такі вхідні змінні, як-от:

- інтенсивність надходження завдань (12 завдань/хв);
- потреба завдань у пам'яті (20–60 сторінок);
- час виконання завдання процесором (нормальний розподіл: середнє 10 с, відхилення 3 с);
- інтенсивність операцій введення-виведення (2–10 с);
- час позиціонування диска (0–0,075 с);
- час передачі даних ($0,001 \times (2,5 + h)$), h у межах 0–25).

1.5 Вихідні змінні

Модель має такі вихідні змінні, як-от:

- загальний час виконання завдань;
- завантаження процесорів, дисків, каналу передачі даних;
- середнє використання пам'яті;
- кількість завдань, що очікують виділення пам'яті чи ресурсів;
- час очікування виділення пам'яті.

1.6 BPMN-діаграма

BPMN-діаграма (Business Process Model and Notation[1]) – це графічне представлення процесів, яке дозволяє детально описати їхню послідовність, взаємодію елементів системи та потік даних між ними. Вона використовується для візуалізації концептуальної моделі, допомагаючи зрозуміти логіку роботи системи, виявити можливі вузькі місця та полегшити подальшу формалізацію і моделювання.

На рисунку 1.1 представлено BPMN-діаграму моделі, яка відображає ключові етапи обробки завдань і взаємодію різних компонентів системи через обмін повідомленнями. Діаграма ілюструє потік даних і послідовність дій від генерації завдання до його завершення.

Розглянемо блок генерації завдань. Через визначений проміжок часу, вказаний у коментарі, відбувається подія створення нового завдання. Після цього завданню призначається обсяг пам'яті (кількість сторінок), необхідний для його обробки, і воно передається в оперативну пам'ять.

Оперативна пам'ять отримує повідомлення про нове завдання, додає його в чергу й перевіряє наявність вільних сторінок для виконання хоча б одного завдання з черги. Щойно вільна пам'ять стає доступною, вона виділяється, і завдання надсилається до процесорів.

Процесори приймають повідомлення про нове завдання і додають його в чергу. Як тільки один із процесорів стає вільним, він починає обробку завдання. По завершенню обробки перевіряються три умови: наявність вільного диска, невиконаний запит введення-виведення та переривання. Коли ці умови виконано, процесор звільняється, а система надсилає кілька повідомлень:

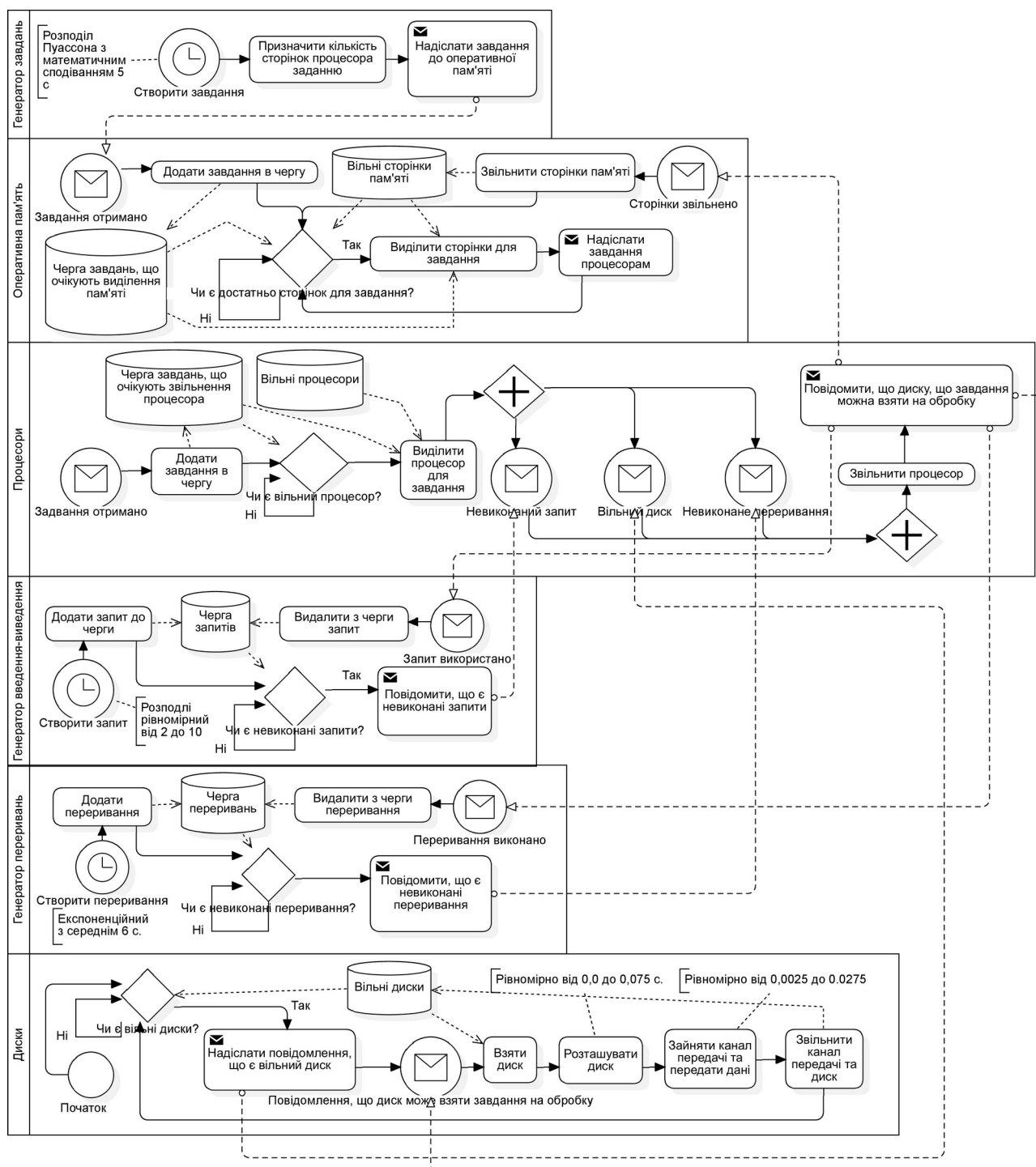


Рисунок 1.1 — BPNM-діаграма

генератору запитів про виконання запиту, оперативній пам'яті про звільнення сторінок і диску про готовність прийняти завдання.

Розглянемо блок генератора запитів. Через заданий час відбувається подія створення нового запиту, який додається в чергу. Далі постійно перевіряється наявність хоча б одного невиконаного запиту в черзі. Якщо такий запит є, повідомлення надсилається процесорам.

Коли диск отримує повідомлення про готовність захопити завдання, він розпочинає позиціонування своєї головки зчитування, після чого виконується передача даних через канал введення-виведення. Завдання вважається завершеним після успішного виконання цих операцій.

1.7 Висновки до розділу

У висновку можна зазначити, що розроблена концептуальна модель багатопроцесорної обчислювальної системи забезпечує базу для формалізації процесів, побудови алгоритму імітації та проведення експериментального дослідження. Визначені вхідні змінні, параметри та вихідні характеристики дозволяють ефективно аналізувати роботу системи та оптимізувати її продуктивність.

2 РОЗРОБКА ФОРМАЛІЗОВАНОЇ МОДЕЛІ

У цьому розділі представлено формалізовану модель багатопроцесорної обчислювальної системи, розроблену з використанням мережі Петрі. Модель описує елементи системи, такі як пам'ять, процесори, диски та канал передачі даних, а також події, що визначають їх взаємодію. Вказано числові параметри для кожного компонента, правила спрацьовування переходів, а також описано генерацію випадкових величин для вхідних змінних. У розділі також подано формули для обчислення вихідних характеристик системи, які дозволяють аналізувати її продуктивність, завантаженість та ефективність роботи. Створена модель є основою для подальшого алгоритму імітації та експериментального дослідження.


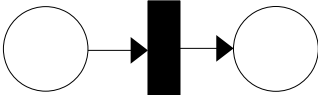



Мережа Петрі – це математичний апарат для моделювання дискретних систем, який складається з позицій, переходів та дуг. Позиції представляють стани системи, переходи – події або дії, що змінюють ці стани, а дуги визначають зв'язки між позиціями і переходами. Стан мережі визначається кількістю маркерів у позиціях, які називаються токенами [2][с. 67].

В основі роботи мережі лежить правило спрацьовування переходу: якщо у всіх вхідних позиціях переходу кількість маркерів не менша за кратність відповідних дуг, перехід активується. Під час спрацьовування маркери переміщуються з вхідних позицій до вихідних відповідно до кратності дуг. Мережа Петрі дозволяє описувати процеси з конфліктними переходами, багатоканальними переходами та стохастичними часовими затримками.

У таблиці 2.1 наведено елементи формалізації мережі Петрі [2][с. 69].

Таблиця 2.1 Мережа Петрі

Назва	Позначення	Опис
Перехід		Позначає подію

Назва	Позначення	Опис
Позиція		Позначає умову
Дуга		Позначає зв'язки між подіями та умовами
Маркер(один)		Позначає виконання (або не виконання) умови
Багато маркерів		Позначає багатократне виконання умови
Багато дуг		Позначає велику кількість зв'язків

2.1 Вхідні параметри

Розглянемо параметри переходів моделі у таблиці 2.2. Нехай маємо відрізок $[M, N]$, що позначає діапазон можливої кількості сторінок, що завдання може займати. Нехай $K \in [M, N]$, де K — кількість сторінок, що займає довільне завдання.

Таблиця 2.2 Параметри переходів

Назва переходу	Часова затримка	Значення пріоритету	Значення ймовірності запуску
generate_task	Poisson(5.0)	0	1

generate_task_K_pages	0	0	$\frac{1}{M-N}$
fail_allocate_task_K_pages	0	-1	1
try_allocate_task_K_pages	0	0	1
wait_allocate_task_20_pages	0	0	1
process_task_K_pages	Norm(10, 3)	$N-K$	1
generate_io_request	Unif(2, 10)	0	1
create_io_task_20_pages	0	$N-K$	1
take_up_disk_task_K_pages	0	$N-K$	1
generate_interrupt_K_pages	Exp(6)	0	1
place_disk_K_pages	Unif(0, 0.075)	$N-K$	1
io_channel_transfer_task_K_pages	Unif(0.0025, 0.0275)	$N-K$	1

Розглянемо параметри позицій у таблиці 2.3.

Таблиця 2.3 Параметри позицій

Назва позиції	Початкове значення
generator_task	1
generated_task	0
task_K_pages	0
fail_allocate_token_task_K_pages	0
allocated_task_K_pages	0
total_wait_allocate_task	0
pages	131

Назва позиції	Початкове значення
generator_io_request	1
processors	2
generated_request	0
io_task_K_pages	0
generator_interrupt	1
generated_interrupt	0
busy_disk_task_K_pages	0
free_disks	4
disk_placed_task_K_pages	0
is_disk_placement_available	1
finished_tasks_task_K_pages	0
finished_tasks	0

2.2 Вихідні параметри

У даному розділі будуть обраховуватися вихідні параметри моделі.

2.2.1 Часи виконання завдання в системі та очікування виділення пам'яті

Визначимо середній час та середнє квадратичне відхилення.

2.2.1.1 Середнє значення

Визначимо середній час за формулою (2.1):

$$\Delta t = \frac{\sum_{n=0}^{N-1} [(t_{\text{exit},n} - t_{\text{enter},n})(t_{\text{exit},n+1} - t_{\text{exit},n})]}{\sum_{n=0}^{N-1} [t_{\text{exit},n+1} - t_{\text{exit},n}]}, \quad (2.1)$$

де n — індекс елемента множин $M_{\text{enter}} = (t_0, t_1, \dots, t_n)$ та $M_{\text{exit}} = (t_0, t_1, \dots, t_n)$, що складаються з точок часу входу та виходу завдання з обробки в системі (очікування виділення пам'яті) відповідно;

N — розмір множин M_{enter} та M_{exit} .

2.2.1.2 Середнє квадратичне відхилення

Визначимо середнє квадратичне відхилення за формулою (2.2):

$$\delta = \sqrt{\frac{\sum_{n=0}^{N-1} [(t_{\text{exit},n} - t_{\text{enter},n}) - \Delta t]^2 (t_{\text{exit},n+1} - t_{\text{exit},n})}{\sum_{n=0}^{N-1} [t_{\text{exit},n+1} - t_{\text{exit},n}]}} \quad (2.2)$$

2.2.2 Кількості завдань в очікуванні пам'яті та зайнятих сторінок

2.2.2.1 Середнє значення

Середня кількість завдань в очікуванні та зайнятих сторінок розраховується однаково за формулою (2.3):

$$P_{\text{average}} = \frac{\sum_{n=0}^{N-1} [(p_n)(t_{n+1} - t_n)]}{\sum_{n=0}^{N-1} [t_{n+1} - t_n]} \quad (2.3)$$

де n — індекс елемента з множини $M = ((p_0, t_0), (p_1, t_1), \dots, (p_n, t_n))$, що позначає пари зі значенням позиції p в момент часу t ;

N — розмір множини M .

2.2.2.2 Середнє квадратичне відхилення

Середнє квадратичне відхилення кількості завдань в очікуванні та зайнятих сторінок розраховується однаково за формулою (2.4):

$$\delta = \sqrt{\frac{\sum_{n=0}^{N-1} [(p_n - p_{\text{average}})^2 (t_{n+1} - t_n)]}{\sum_{n=0}^{N-1} [t_{n+1} - t_n]}}, \quad (2.4)$$

2.2.3 Завантаження процесорів, дисків та каналу передачі

Завантаження в момент часу t процесорів, дисків та каналу передачі обчислюється аналогічно за формулою (2.5).

$$l = \frac{\sum_{n=N_{\min}}^{N_{\max}} [t_{\text{service}}[n]]}{t}, \quad (2.5)$$

де n — це елемент N , що складається з натуральних чисел $[N_{\min}, N_{\max}]$, які позначають кількість сторінок, яку може займати завдання;

t_{service} — це відображення $N \rightarrow M$, де $M = \{t_k\}_{k=0}^{N_{\max}-N_{\min}}$, що позначає сукупність переходів, які стосуються або процесорів, або дисків, або каналу передачі.

Однак за умови, що K — кількість пристроїв, що обслуговують, — більша одиниці, і кожен з них може працювати увесь час, то навантаження буде в межах від $[0, K]$.

Тому розрахуємо усереднене навантаження для одного процесора (диска) за формулою (2.6):

$$\bar{l} = \frac{l}{K} \quad (2.6)$$

2.2.3.1 Середнє значення

Знаючи навантаження в момент часу t , можемо обчислити середнє значення навантаження за формулою (2.7):

$$l_{average} = \frac{\sum_{n=0}^{N-1} [(\bar{l}_n)(t_{n+1} - t_n)]}{\sum_{n=0}^{N-1} [t_{n+1} - t_n]}, \quad (2.7)$$

де n — індекс елемента з множини $M = ((l_0, t_0), (l_1, t_1), \dots, (l_n, t_n))$, що позначає пари зі значенням навантаження l в момент часу t ;

N — розмір множини M .

2.2.3.2 Середнє квадратичне відхилення

Середнє квадратичне відхилення розраховується за формулою (2.8):

$$\delta = \sqrt{\frac{\sum_{n=0}^{N-1} [(\bar{l}_n - \bar{l}_{average})^2 (t_{n+1} - t_n)]}{\sum_{n=0}^{N-1} [t_{n+1} - t_n]}}, \quad (2.8)$$

2.3 Мережа Петрі

На рисунку 2.1 можна розглянути діаграму мережі Петрі. Для зручності вона побудована лише з двома типами завдань. Це робить її менш нагромадженою, оскільки частини схеми для кожного завдання є однаковими.

Основними компонентами мережі є генератор завдань, оперативна пам'ять, процесори, генератор запитів введення-виведення, диски, канал передачі даних та блок завершення роботи.

Генератор завдань представлений позицією `generated_task`, яка містить токен, що активує перехід `generate_task`. Цей перехід генерує нові завдання з пуассонівським розподілом часу. Згенеровані завдання додаються в позиції, які відповідають їхньому обсягу пам'яті, наприклад, `task_20_pages` для завдань із потребою у 20 сторінок.

Оперативна пам'ять представлена позицією `pages`, яка відображає загальну кількість доступних сторінок. Переходи `try_allocate` і `wait_allocate` перевіряють наявність вільної пам'яті та виділяють її для завдань відповідного

обсягу. У разі успішного виділення токени передаються до позиції `allocated`, а у разі невдачі – у `fail_allocate_token`.

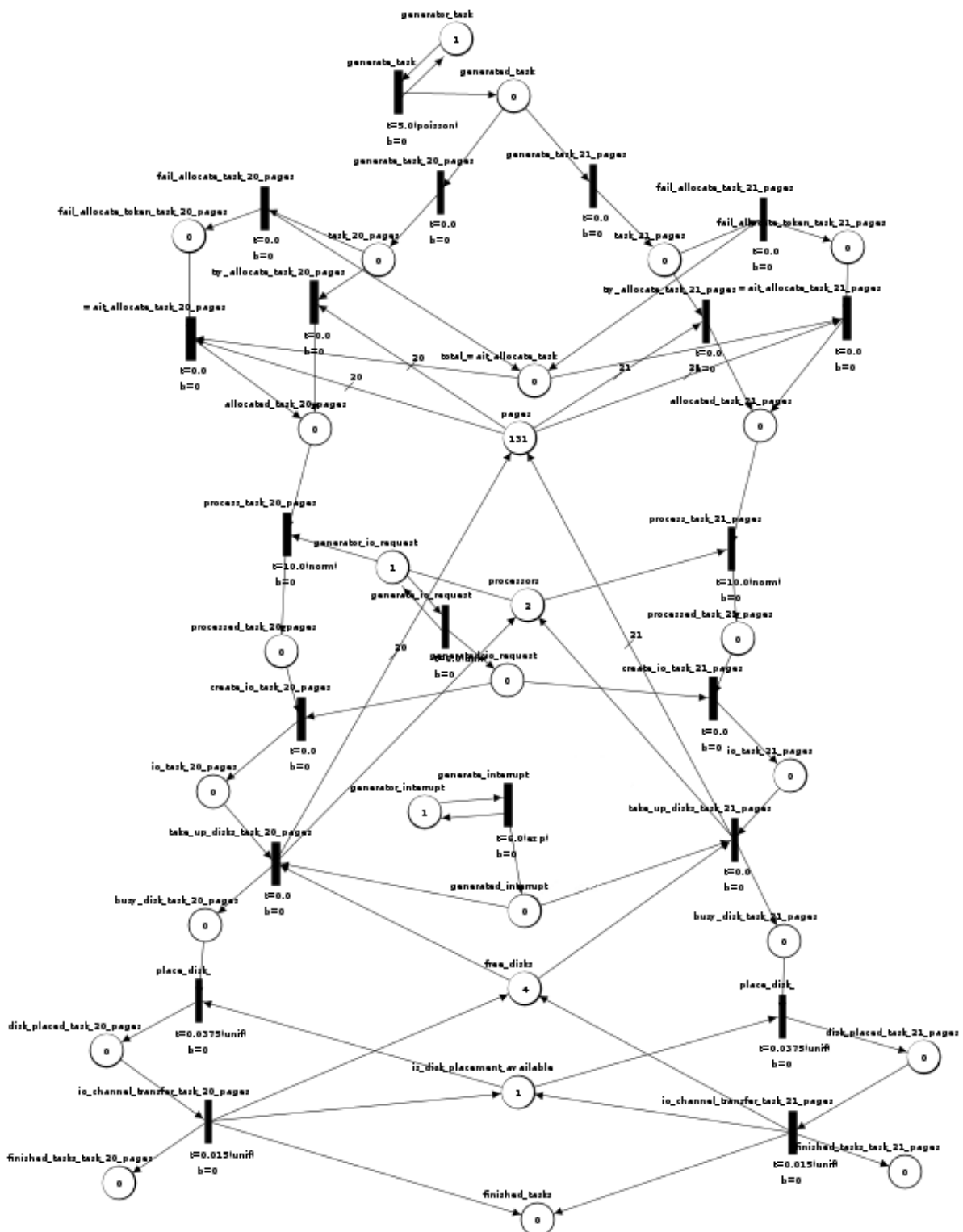


Рисунок 2.1 — Мережа Петрі з двома типами завдань

Процесори моделюються позицією `processors`, яка визначає кількість доступних процесорів. Переходи `process` відповідають за обробку завдань із використанням нормального розподілу часу.

Генератор запитів введення-виведення моделюється позицією `generated_io_request`, яка активує перехід `create_io`. Цей перехід генерує запити введення-виведення для завдань, які були оброблені процесорами.

Диски представлені позицією `free_disks`, яка відображає кількість доступних дисків (4 диски). Перехід `place_disk` імітує позиціонування головки зчитування з використанням рівномірного розподілу часу. Після виконання цієї операції токени передаються через канал введення-виведення.

Канал введення-виведення моделюється позицією `is_disk_placement_available`, яка визначає готовність каналу до передачі даних. Перехід `io_channel_transfer` відповідає за передачу даних із використанням рівномірного розподілу часу. Завершення роботи моделюється позицією `finished_tasks`, яка накопичує виконані завдання. Це дозволяє аналізувати продуктивність системи та обчислювати необхідні показники.

2.4 Висновки до розділу

У цьому розділі було створено формалізовану модель багатопроцесорної обчислювальної системи, яка дозволяє математично описати її компоненти та взаємодії. Модель побудована з використанням мережі Петрі, що забезпечує можливість детального моделювання процесів генерації, обробки завдань, запитів введення-виведення та управління ресурсами. Було визначено всі необхідні вхідні параметри, їхні числові характеристики, а також формули для обчислення вихідних параметрів. Створена модель є основою для розробки алгоритму імітації та проведення експериментального дослідження роботи системи.

3 АЛГОРИТМІЗАЦІЯ МОДЕЛІ ТА ЇЇ РЕАЛІЗАЦІЯ

Цей розділ присвячений алгоритмізації та програмній реалізації імітаційної моделі багатопроцесорної обчислювальної системи, створеної за допомогою мережі Петрі. Метою є опис роботи стандартного алгоритму моделювання, наданого бібліотекою PetriObjModelPaint [3], а також внесених модифікацій, які забезпечують відповідність алгоритму вимогам поставленої задачі. Розділ містить опис основних етапів алгоритму, реалізацію додаткового функціоналу, оптимізацію коду під конкретну задачу, а також протокол подій, що фіксуються під час моделювання. Наприкінці представлено результати верифікації моделі, що підтверджують коректність її реалізації.

3.1 Опис алгоритму імітації мережі Петрі

Алгоритм імітації мережі Петрі базується на послідовному просуванні моделі в часі та обробці подій, які відбуваються у системі. Основна мета алгоритму — забезпечити виконання правил роботи мережі Петрі, таких як спрацювання переходів за наявності необхідних умов та обчислення статистичних характеристик позицій і переходів.

Алгоритм визначає мінімальний час запуску переходу (eventMin) і просуває глобальний час моделі до цього моменту. Це забезпечує обробку подій у правильному хронологічному порядку.

На кожному етапі перевіряються активні переходи. Якщо перехід може спрацювати (умови виконуються), він захоплює необхідні маркери з пов'язаних позицій. Конфлікти між переходами вирішуються за пріоритетами або ймовірностями.

Після запуску перехід розподіляє маркери по вихідних позиціях. Якщо перехід має накопичувальний буфер, процес виходу маркерів може повторюватися, поки буфер не стане порожнім.

Алгоритм збирає статистичні дані для кожної позиції і переходу: середню кількість маркерів у позиціях і середній час обробки переходів.

Якщо кілька переходів можуть спрацювати одночасно, вибір здійснюється на основі пріоритетів або ймовірностей. Це забезпечує гнучкість і відповідність алгоритму до стохастичної природи системи.

Розглянемо псевдокод алгоритму нижче. Функція `go` виконує ітерації, поки поточний час не досягне часу моделювання. На кожній ітерації обчислюються статистичні показники, відбувається перехід до часу наступної події, а також виконується обробка поточного переходу.

Функція `input` шукає активні переходи, які можуть спрацювати. Виконує вирішення конфліктів між переходами, які мають однаковий пріоритет.

Функція `output` Обробляє спрацьовування переходів і розподіл маркерів у відповідних позиціях. Виконує додаткові перевірки для переходів із накопичувальними буферами, щоб повністю обробити всі події, які можуть відбутися в поточний момент часу.

```

1 def go():
2     # Initialize simulation time and current time
3     set_simulation_time(time_modelling)
4     set_current_time(0)
5     # Initialize transitions and positions
6     input() # Activate initial transitions
7     # Main simulation loop
8     while current_time < simulation_time:
9         # Update statistics for the current time period
10        do_statistics()
11        # Move to the next event's time
12        set_current_time(get_time_min())
13        # Log event details (if logging is enabled)
14        if logging_enabled:
15            log("Time:", current_time, "; Event:", event_min.get_name())
16
17        # Process the event and its effects
18        output()
19        # Reactivate transitions after event execution

```

```

20     input()
21
22 def input():
23     # Find all active transitions
24     active_transitions = find_active_transitions()
25     # If no transitions are active and all buffers are empty, stop further processing
26     if active_transitions is empty and all_buffers_are_empty():
27         set_time_min(infinity)
28     else:
29         # While there are active transitions
30         while active_transitions is not empty:
31             # Resolve conflicts between active transitions
32             selected_transition = resolve_conflicts(active_transitions)
33             # Activate the selected transition
34             selected_transition.act_in(listP, current_time)
35             # Update the list of active transitions
36             active_transitions = find_active_transitions()
37             # Update the minimum time for the next event
38             find_event_min()
39
40 def output():
41     # Ensure that the simulation time has not been exceeded
42     if current_time <= simulation_time:
43         # Execute the event with the minimum time
44         event_min.act_out(listP, current_time)
45
46     # Handle remaining events in the transition's buffer
47     while event_min.has_buffer():
48         event_min.update_event_time()
49         if event_min.get_min_time() == current_time:
50             event_min.act_out(listP, current_time)
51         else:
52             break
53     # Check other transitions for events occurring at the current time

```

```

54     for transition in listT:
55         if transition.has_buffer() and transition.get_min_time() == current_time:
56             transition.act_out(listP, current_time)
57             # Process additional events in the buffer
58         while transition.has_buffer():
59             transition.update_event_time()
60             if transition.get_min_time() == current_time:
61                 transition.act_out(listP, current_time)
62             else:
63                 break

```

3.2 Модифікації алгоритму

У процесі реалізації моделі були внесені зміни до базового алгоритму, щоб адаптувати його до специфіки поставленої задачі. Основні модифікації стосувалися покращення роботи мережі Петрі та збору додаткової інформації для аналізу. Зміни можна поділити на два напрямки: функціональні вдосконалення алгоритму та зручність роботи з мережами.

3.2.1 Додавання розподілу Пуассона для генерації подій

Для точного моделювання вхідних потоків завдань у систему було реалізовано підтримку розподілу Пуассона. Це дозволило описати інтенсивність надходження завдань у вигляді стохастичного процесу, що відповідає заданій середній інтенсивності. Реалізація виконана шляхом додавання відповідної функції до генератора подій у мережі.

```

1  public static double poisson(final double timeMean) {
2      PoissonDistribution poisson = new PoissonDistribution(timeMean);
3      return poisson.sample();
4  }

```

3.2.2 Вдосконалення роботи з NetLibrary

Для зручності роботи з мережами Петрі було додано клас NetLibraryManager, який дозволяє динамічно додавати методи в бібліотеку

NetLibrary. Завдяки цьому можна легко інтегрувати нові мережі в графічний інтерфейс, використовувати рефлексію для виклику методів, і забезпечувати їхню коректність під час компіляції. Також додано перевірку методів генерації мережі за допомогою анотацій NetLibraryMethod та NetLibraryMethodProcessor.

```

1 @NetLibraryMethod
2 public static PetriNet createNet() throws ExceptionInvalidTimeDelay {
3     final CourseWorkNet net = new CourseWorkNet();
4     return net.net;
5 }

```

3.2.3 Покращення графічного інтерфейсу

Був виправлений метод generateGraphNetBySimpleNet, що забезпечує коректне графічне відображення мережі Петрі. Помилка полягала у тому, що оскільки не було перевірки на включення переходів та позицій в контейнери inTrans та inPlaces, то дублікати додавалися у графічне відображення мережі.

```

1 if (
2     !inTrans.contains(tran) // Нова перевірка
3     && tran.getNumber() == outArc.getNumT()
4 ) {
5     inTrans.add(tran);
6 }
7 // ...
8 if (
9     !inPlaces.contains(place) // Нова перевірка
10    && place.getNumber() == inArc.getNumP()
11 ) {
12    inPlaces.add(place);
13 }

```

3.2.4 Клас CourseWorkPetriSim

Клас CourseWorkPetriSim реалізує механізм імітації роботи мережі Петрі, дозволяючи відстежувати динаміку її роботи протягом заданого часу

моделювання. Він включає методи для управління часом, обробки подій, вирішення конфліктів між переходами та збору статистичних даних.

3.2.4.1 Опис атрибутів

Одним з ключових атрибутів є `timeState` - об'єкт класу `StateTime`, який зберігає інформацію про поточний час моделювання (`currentTime`) та час завершення моделювання (`simulationTime`). Він є центральним для управління часом у моделі, дозволяючи просувати глобальний час до наступної події.

Атрибут `timeMin` містить мінімальний час спрацювання серед усіх переходів у мережі. Цей атрибут оновлюється під час виконання моделі, щоб визначити найближчу подію для обробки.

У класі також присутній масив `listP`, що містить усі позиції мережі Петрі (`PetriP`). Кожен елемент описує стан позиції, зокрема кількість маркерів у ній. Цей масив використовується для перевірки умов активації переходів і переміщення маркерів.

Важливим компонентом є масив `listT`, що містить усі переходи мережі Петрі (`PetriT`). У ньому зберігається інформація про затримки, умови спрацювання, ймовірності, пріоритети, буфери та час наступного спрацювання кожного переходу.

Останнім ключовим атрибутом є `eventMin` - об'єкт типу `PetriT`, що представляє перехід із мінімальним часом спрацювання. Цей атрибут визначає, який перехід буде оброблено наступним.

3.2.4.2 Опис методів

Конструктор класу `CourseWorkPetriSim` приймає об'єкт `PetriNet` як вхідний параметр. Під час ініціалізації встановлюються початкові значення часу моделювання, список позицій (`listP`) та список переходів (`listT`), які беруться з переданої мережі. Також викликається метод `getEventMin()`, який визначає перший перехід із мінімальним часом затримки. Це дозволяє налаштувати початковий стан моделі для початку імітації.

Основний метод класу, `go`, виконує імітацію протягом заданого часу моделювання. Він приймає параметр `timeModelling`, що визначає загальну тривалість симуляції, та `trackStats` — функцію, яка дозволяє збирати статистичні дані під час виконання. Метод ініціалізує початковий час моделювання, потім ітеративно викликає методи `input` і `output` для обробки подій. В кожному кроці часу оновлюється поточний час, імітуються активація та завершення переходів, а також збирається інформація для подальшого аналізу.

Метод `eventMin` використовується для визначення переходу з мінімальним часом затримки серед усіх доступних. Він переглядає список переходів і знаходить той, що має найменший час. Цей метод також оновлює значення `timeMin` і визначає поточний активний перехід `eventMin`, який буде виконуватись наступним.

Метод `findActiveT` відповідає за пошук усіх активних переходів у мережі. Він перевіряє кожен перехід на відповідність умовам, що визначені у зв'язках між позиціями, та додає його до списку активних, якщо ймовірність його спрацьовування більше нуля. У випадку, якщо кілька переходів одночасно задовольняють умови, вони сортуються за пріоритетом.

Обробка конфліктів між активними переходами здійснюється за допомогою методу `doConflict`. У разі, якщо кілька переходів мають однаковий пріоритет, вибір між ними здійснюється випадковим чином із урахуванням ймовірностей, визначених для кожного переходу. Цей механізм гарантує, що імітація враховує стохастичну природу процесів у системі.

Методи `input` та `output` відповідають за обробку подій у мережі. Метод `input` ідентифікує активні переходи та виконує їх активацію, якщо умови дозволяють це зробити. Він також визначає, коли немає жодного активного переходу або всі буфери переходів порожні, встановлюючи час моделювання в максимальне значення. Метод `output` обробляє завершення переходів і переміщення маркерів між позиціями, а також виконує спрацьовування переходів, якщо в їх буферах залишаються маркери.

Методи для отримання та встановлення часу, такі як `getTimeMin`, `setTimeCurr`, `getSimulationTime`, та `setSimulationTime`, забезпечують точний контроль за поточним часом моделювання та дозволяють синхронізувати всі компоненти мережі.

3.2.5 Клас **CourseWorkNet**. Протокол подій

Клас `CourseWorkNet` розроблений для аналізу та моделювання роботи багатопроцесорної обчислювальної системи за допомогою мережі Петрі. Він є обгорткою для створення та налаштування компонентів мережі, забезпечуючи зручний інтерфейс для опису та управління моделлю.

3.2.5.1 Опис атрибутів

Позиція `generated_task` зберігає кількість згенерованих завдань у системі. Ця позиція використовується для активізації переходів, що створюють нові завдання. Її основне призначення полягає у фіксації вхідного потоку завдань до системи.

Позиція `generated_io_request` відповідає за генерацію запитів введення-виведення для завдань після їх обробки. Її призначення полягає в забезпеченні інтеграції завдань із підсистемою дисків.

Позиція `generated_interrupt` використовується для моделювання переривань, які запускають запити до дисків. Її призначення - імітувати реальні події введення-виведення у багатозадачній системі.

Позиція `processors` відображає кількість вільних процесорів у системі, яка початково становить два. Вона використовується для визначення доступності процесорів для обробки завдань.

Позиція `pages` відображає кількість доступних сторінок пам'яті в системі, початково маючи 131 сторінку. Її призначення полягає у визначенні можливості виділення пам'яті для завдань.

Позиція `free_disks` відображає кількість доступних дисків, яких початково чотири. Вона забезпечує розподіл завдань між доступними дисками.

Позиція `total_wait_allocate_task` підраховує загальний час очікування завдань на виділення пам'яті. Вона використовується для аналізу затримок у системі.

Позиція `finished_tasks` зберігає кількість завершених завдань. Її призначення полягає у відображенні загальної продуктивності системи.

Позиція `is_disk_placement_available` вказує на доступність каналу введення-виведення для передачі даних. Вона контролює одночасний доступ до каналу.

Перехід `generate` генерує нові завдання у системі із заданими параметрами, такими як обсяг пам'яті. Його призначення полягає у забезпеченні створення потоку завдань.

Перехід `try_allocate` перевіряє наявність вільної пам'яті для завдань. У разі успіху пам'ять виділяється. Його призначення - запускати обробку завдань, які отримали доступ до пам'яті.

Перехід `fail_allocate` відображає спробу виділення пам'яті, яка завершилась невдачею. Його призначення полягає у поверненні завдання у чергу для подальших спроб.

Перехід `wait_allocate` переводить завдання з черги у стан обробки після виділення пам'яті. Цей перехід забезпечує повторне надання пам'яті для завдань.

Перехід `process` імітує обробку завдань процесорами із використанням нормального розподілу часу. Це основний перехід для виконання завдань.

Перехід `create_io` створює запити введення-виведення для завдань, які були оброблені. Його призначення полягає в інтеграції із підсистемою дисків.

Перехід `take_up_disks` захоплює доступний диск для обробки запиту. Він контролює завантаження дисків.

Перехід `place_disk` імітує позиціонування головки диска перед операцією. Його призначення - затримувати виконання запиту для моделювання реального часу доступу.

Перехід `io_channel_transfer` імітує передачу даних через канал введення-виведення. Цей перехід завершує обробку запитів введення-виведення.

3.2.5.2 Опис TaskObject

Об'єкт `TaskObject` представляє завдання з конкретним обсягом пам'яті (від 20 до 60 сторінок) та визначеним життєвим циклом у системі. Цей об'єкт містить низку важливих полів, що керують його функціонуванням у системі.

Поле `generate` є генератором завдання цього типу. Поле `task` являє собою позицію, що зберігає завдання. `Try_allocate` є переходом для перевірки доступності пам'яті, тоді як поле `allocated` - це позиція, яка фіксує завдання, що отримали пам'ять.

`Fail_allocate` є переходом, що фіксує невдалі спроби виділення пам'яті. Позиція `fail_allocate_token` призначена для невдалих завдань, які очікують повторного виділення пам'яті. `Wait_allocate` представляє собою перехід для повторного виділення пам'яті.

`Process` є переходом, що імітує обробку завдання, а `create_io` - це перехід, який створює запити введення-виведення. `Take_up_disks` виступає переходом для захоплення дисків завданням.

Поле `busy_disk` є позицією, яка показує, що диск зайнятий завданням. `Place_disk` представляє перехід, що імітує позиціонування диска, а `disk_placed` - це позиція, яка фіксує успішне позиціонування диска.

`Io_channel_transfer` є переходом, що забезпечує передачу даних, а `finish` - це позиція, яка зберігає завершені завдання.

Об'єкти `TaskObject` використовуються для відстеження стану кожного типу завдань окремо, включаючи їх генерацію, обробку, введення-виведення та завершення. Завдяки цьому можна проводити детальний аналіз роботи системи та оптимізувати її параметри.

3.2.5.3 Опис методів

Конструктор `CourseWorkNet` є основним конструктором класу, який створює та ініціалізує всі компоненти мережі Петрі. Він створює генератори

завдань, запитів введення-виведення та переривань, позиції для обробки завдань, пам'яті, процесорів, дисків, а також переходи для роботи з ресурсами. Додаткова логіка включає генерацію завдань з різними параметрами, розрахунок ймовірностей для кожного типу завдання та додавання дуг між позиціями та переходами.

Метод `create_task_generator` створює генератор завдань, який додає нові завдання до системи відповідно до розподілу Пуассона. Він приймає параметри `d_P` (список позицій), `d_T` (список переходів), `d_In` (список вхідних дуг) та `d_Out` (список вихідних дуг). В результаті повертає позицію `generated_task`, яка накопичує створені завдання.

Метод `create_processors` створює позицію `processors`, яка відображає кількість доступних процесорів. Він приймає єдиний параметр `d_P` (список позицій) та повертає позицію `processors`.

Метод `create_pages` створює позицію `pages`, що відображає кількість доступних сторінок пам'яті. Приймає параметр `d_P` (список позицій) та повертає позицію `pages`.

Метод `create_free_disks` створює позицію `free_disks`, яка відповідає кількості доступних дисків. Приймає параметр `d_P` (список позицій) та повертає позицію `free_disks`.

Метод `create_io_request_generator` створює генератор запитів введення-виведення з рівномірним розподілом часу. Він приймає параметри `d_P` (список позицій), `d_T` (список переходів), `d_In` (список вхідних дуг) та `d_Out` (список вихідних дуг). В результаті повертає позицію `generated_io_request`.

Метод `create_interrupt_generator` створює генератор переривань із експоненційним розподілом часу. Він приймає параметри `d_P` (список позицій), `d_T` (список переходів), `d_In` (список вхідних дуг) та `d_Out` (список вихідних дуг). В результаті повертає позицію `generated_interrupt`.

Метод `generate_task_objects` створює об'єкти завдань `TaskObject` для кожного обсягу пам'яті між `pages_start` та `pages_end` із заданою ймовірністю. Він приймає параметри `pages_start` (мінімальний обсяг пам'яті завдання), `pages_end`

(максимальний обсяг пам'яті завдання) та *probability* (ймовірність створення завдання кожного типу). В результаті додає об'єкти *TaskObject* до списку *taskObjects*.

3.3 Верифікація

У цьому підрозділі опишемо проведення верифікації моделі, класи для збору аналізу даних, приклади коду. Покажемо, що модель досягає сталих повторюваних середніх значень.

3.3.1 Збір даних. Клас *GatherDataCourseWorkNet*

Клас *GatherDataCourseWorkNet* призначений для верифікації моделі та збору статистичних даних. Він дозволяє багаторазово запускати імітаційну модель, збирати характеристики (завантаження ресурсів, час виконання завдань, використання пам'яті) та записувати результати у файли для аналізу.

Основні компоненти:

- *DiffTimePoint* – зберігає часові точки та різниці між ними для аналізу часу перебування завдань у системі та очікування ресурсів;
- *PropertyStats* – фіксує ключові характеристики: завантаження ресурсів, часи моделювання, використання пам'яті, кількість завдань в очікуванні.

Основні методи:

- *collectStats* – проводить один прогін моделі, збираючи дані про її поведінку;
- *writeStats* – забезпечує серію прогонів, результати яких записуються у файли (*commonProps.csv*, *timeInSystem.csv*, *timeWaitAllocate.csv*);
- *sortDiffTimePointArray* – сортує часові дані;

VerifyModel – верифікує модель за кількома наборами параметрів, записуючи результати в окремі директорії.

Загалом час моделювання складає 800000 одиниць. Кількість прогонів 5 для кожного набору параметрів. Вхідні параметри описані в таблиці А.1.

```

1 static class VerifyModel {
2   public static void main(String[] args) throws IOException {
3     final int[] pagesNums    = new int[] { 131, 131, 200, 400, 700, 1000, 1000};
4     final int[] processorsNums = new int[] { 2, 2, 4, 5, 12, 40, 30};
5     final int[] diskNums      = new int[] { 4, 4, 5, 11, 12, 8, 30};
6     final int[] pagesStarts   = new int[] { 20, 20, 30, 70, 30, 60, 70};
7     final int[] pagesEnds     = new int[] { 60, 60, 40, 100, 70, 100, 80};
8     final int[] tasksTimeMeans = new int[] { 5, 7, 8, 8, 8, 9, 15};
9
10    for (int i = 0; i < pagesEnds.length; i++) {
11      final int pagesNum = pagesNums[i];
12      final int processorsNum = processorsNums[i];
13      final int diskNum = diskNums[i];
14      final int pagesStart = pagesStarts[i];
15      final int pagesEnd = pagesEnds[i];
16      final int tasksTimeMean = tasksTimeMeans[i];
17
18      String dirName = String.format("%d_%d_%d_%d_%d_%d", pagesNum,
processorsNum, diskNum, pagesStart, pagesEnd, tasksTimeMean);
19      File directory = new File(dirName);
20      if (!directory.exists()) {
21        directory.mkdir();
22      }
23      dirName += "/";
24
25      writeStats(dirName + "commonProps.csv", dirName + "timeInSystem.csv", dirName +
"timeWaitAllocate.csv", 5, 800000, pagesNum, processorsNum, diskNum, pagesStart, pagesEnd,
tasksTimeMean);
26    }
27  }
28 }

```

Файл `commonProps.csv` містить загальні характеристики моделі для кожної часової точки симуляції в кожному запуску. Формат даних:

- `runNumber` – номер прогону моделювання;

- `timePoint` – момент часу в процесі моделювання;
- `diskLoad` – завантаження дисків у відсотках (розраховується як середнє завантаження дисків у момент `timePoint`);
- `ioChannelLoad` – завантаження каналу передачі даних (у відсотках);
- `processorsLoad` – завантаження процесорів (у відсотках);
- `totalWaitAllocate` – кількість завдань, що очікують виділення ресурсів;
- `useOfPage` – кількість сторінок пам'яті, що використовується.

Файл `timeInSystem.csv` файл містить дані про час перебування завдань у системі. Формат даних:

- `runNumber` – номер прогону моделювання;
- `timePoint` – момент часу виходу завдання із системи;
- `timeInSystem` – час, проведений завданням у системі (від моменту генерації до завершення всіх операцій).

Файл `timeWaitAllocate.csv` зберігає інформацію про час очікування завдань у черзі на виділення ресурсів. Формат даних:

- `runNumber` – номер прогону моделювання;
- `timePoint` – момент часу, коли завдання завершило очікування ресурсу;
- `timeWaitAllocate` – час, проведений завданням у черзі на виділення ресурсів.

3.3.2 Обробка даних

Для обробки даних застосуємо мову програмування Python [4]. Python пропонує широкий спектр підходів до аналізу та обробки даних, які охоплюють різні аспекти роботи з інформацією: від попередньої обробки до візуалізації та машинного навчання. Для того щоб обробити дані, застосуємо сторонні бібліотеки Pandas [5] та NumPy [6].

Pandas — це бібліотека для обробки та аналізу структурованих даних. Вона забезпечує зручну роботу з двома основними структурами: `DataFrame` (таблиця з рядками та стовпцями) і `Series` (одновимірний масив даних). Pandas дозволяє легко завантажувати дані з різних джерел (CSV, Excel, SQL, JSON),

очищати їх, виконувати групування, фільтрацію, обчислювати статистичні характеристики та працювати з часовими рядами.

NumPy — це бібліотека для ефективної роботи з числовими даними. Її основна структура — `ndarray` (мультивимірний масив), який забезпечує швидкі математичні операції над великими масивами. NumPy також пропонує інструменти для роботи з лінійною алгеброю, генерації випадкових чисел і інтеграції з іншими бібліотеками для обчислень.

Нижче опишемо функції розрахунку середнього значення та середнього квадратичного відхилення відповідно до пункту 2.2.

```

1 import numpy as np
2 import pandas as pd
3
4 def calculate_mean(time_points: pd.Series, values: pd.Series) -> float:
5     prev_time_point = time_points.iloc[0]
6     delay_sum = 0.0
7     value_sum = 0.0
8
9     for time_point, value in zip(time_points.iloc[1:], values):
10         delay = time_point - prev_time_point
11         prev_time_point = time_point
12         delay_sum += delay
13         value_sum += value * delay
14
15     try:
16         res = value_sum / delay_sum
17         return res
18     except ZeroDivisionError:
19         return 0
20
21 def calculate_std_dev(time_points: pd.Series, values: pd.Series, mean: float) -> float:
22     prev_time_point = time_points.iloc[0]
23     delay_sum = 0.0
24     value_sum = 0.0

```

```

25
26 for time_point, value in zip(time_points.iloc[1:], values):
27     delay = time_point - prev_time_point
28     prev_time_point = time_point
29     delay_sum += delay
30     value_sum += ((value - mean) ** 2) * delay
31
32 try:
33     res = np.sqrt(value_sum / delay_sum)
34     return res
35 except ZeroDivisionError:
36     return 0

```

Нижче наведено використання цих функцій під час обчислення статистичних характеристик. Результати наведені у таблицях А.2 та А.3.

```

1 from array import array
2
3 @attr.frozen
4 class MeanStddevStats:
5     diskLoad_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
6     diskLoad_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
7     ioChannelLoad_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
8     ioChannelLoad_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
9     processorsLoad_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
10    processorsLoad_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
11    totalWaitAllocate_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
12    totalWaitAllocate_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
13    useOfPage_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
14    useOfPage_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
15    timeInSystem_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
16    timeInSystem_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
17    timeWaitAllocate_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
18    timeWaitAllocate_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
19
20 mean_stddev_stats_list: list[pd.DataFrame] = []

```

```

21
22 for index, params_data in enumerate(datas):
23     mean_stddev_stats = MeanStddevStats()
24
25     for run_num, group in params_data.common_props.groupby('runNumber'):
26         # Calculate means and standard deviations
27         diskLoad_mean = calculate_mean(group['timePoint'], group['diskLoad'])
28         diskLoad_std_dev = calculate_std_dev(group['timePoint'], group['diskLoad'],
diskLoad_mean)
29
30         ioChannelLoad_mean = calculate_mean(group['timePoint'], group['ioChannelLoad'])
31         ioChannelLoad_std_dev = calculate_std_dev(group['timePoint'], group['ioChannelLoad'],
ioChannelLoad_mean)
32
33         processorsLoad_mean = calculate_mean(group['timePoint'], group['processorsLoad'])
34         processorsLoad_std_dev = calculate_std_dev(group['timePoint'], group['processorsLoad'],
processorsLoad_mean)
35
36         totalWaitAllocate_mean = calculate_mean(group['timePoint'], group['totalWaitAllocate'])
37         totalWaitAllocate_std_dev = calculate_std_dev(group['timePoint'],
group['totalWaitAllocate'], totalWaitAllocate_mean)
38
39         useOfPage_mean = calculate_mean(group['timePoint'], group['useOfPage'])
40         useOfPage_std_dev = calculate_std_dev(group['timePoint'], group['useOfPage'],
useOfPage_mean)
41
42         mean_stddev_stats.diskLoad_mean.append(diskLoad_mean)
43         mean_stddev_stats.diskLoad_std_dev.append(diskLoad_std_dev)
44
45         mean_stddev_stats.ioChannelLoad_mean.append(ioChannelLoad_mean)
46         mean_stddev_stats.ioChannelLoad_std_dev.append(ioChannelLoad_std_dev)
47
48         mean_stddev_stats.processorsLoad_mean.append(processorsLoad_mean)
49         mean_stddev_stats.processorsLoad_std_dev.append(processorsLoad_std_dev)

```

```

50
51     mean_stddev_stats.totalWaitAllocate_mean.append(totalWaitAllocate_mean)
52     mean_stddev_stats.totalWaitAllocate_std_dev.append(totalWaitAllocate_std_dev)
53
54     mean_stddev_stats.useOfPage_mean.append(useOfPage_mean)
55     mean_stddev_stats.useOfPage_std_dev.append(useOfPage_std_dev)
56
57     for run_num, group in params_data.time_in_system.groupby('runNumber'):
58         timeInSystem_mean = calculate_mean(group['timePoint'], group['timeInSystem'])
59         timeInSystem_std_dev = calculate_std_dev(group['timePoint'], group['timeInSystem'],
timeInSystem_mean)
60         mean_stddev_stats.timeInSystem_mean.append(timeInSystem_mean)
61         mean_stddev_stats.timeInSystem_std_dev.append(timeInSystem_std_dev)
62
63     for run_num, group in params_data.time_wait_allocate.groupby('runNumber'):
64         timeWaitAllocate_mean = calculate_mean(group['timePoint'], group['timeWaitAllocate'])
65         timeWaitAllocate_std_dev = calculate_std_dev(group['timePoint'],
group['timeWaitAllocate'], timeWaitAllocate_mean)
66         mean_stddev_stats.timeWaitAllocate_mean.append(timeWaitAllocate_mean)
67         mean_stddev_stats.timeWaitAllocate_std_dev.append(timeWaitAllocate_std_dev)
68
69     dt = pd.DataFrame(attr.asdict(mean_stddev_stats))
70     dt['params_index'] = index
71     mean_stddev_stats_list.append(dt)

```

Нижче розрахуємо глобальні середні значення та квадратичні відхилення, а також відсоткові відхилення значень кожного прогону відносно них. Результати можна розглянути в таблицях А.4, А.5, А.6, А.7 відповідно.

```

1 for i, mean_stddev_stats in mean_stddev_stats_data_frame.groupby('params_index'):
2     means = mean_stddev_stats.mean()
3     global_mean_stddev_list.append(means)
4     mean_stddev_stats_relative_mean = ((mean_stddev_stats - means).abs() * 100) / means
5     mean_stddev_stats_relative_mean.fillna(0, inplace=True)
6     mean_stddev_stats_relative_mean['params_index'] = i

```

```
7 mean_stddev_stats_relative_mean_list.append(mean_stddev_stats_relative_mean)
```

3.3.3 Аналіз даних

Після збору та обробки даних перейдемо до їхнього аналізу.

3.3.3.1 Середні значення

З таблиці А.2 бачимо аномально високі значення для набору 6. Для зручності їх наведено в таблиці 3.1. Це можна пояснити тим, що час обробки завдання більший ніж період між надходженнями нових завдань, а також час моделювання великий 800000 вони й досягають таких високих значень.

Таблиця 3.1 Аномально високі значення набору 6

Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
13380.9207081260 88	113.311319697821 83	3308.74926877018 4	4594.38503977522 3
13780.9840818338 49	113.226667153069 29	2813.80493761689 24	4028.65343939817 06
13257.5187966601 3	113.210901879149 79	3044.63032273181 76	4279.44274629576 2
13564.4116141748 62	112.767507004493 82	3461.19579759108 14	4760.63651209371 6
13360.9585336715 77	112.665021434405 32	2532.94246609069 2	3496.31099089404 05

Розглянемо таблиці А.4 та А.6, проаналізуємо відсоткові відхилення середніх значень відносно глобального середнього для кожного набору даних у таблиці 3.2.

Таблиця 3.2 Аналіз відхилень середніх значень відносно глобальних

Індекс набору параметрів	Аналіз
0	Система працює стабільно, і всі відхилення менші 2%.
1	Бачимо, що кількість завдань в очікуванні пам'яті варіюється і доходить до 67%. Час виділення пам'яті доростає до 141%. Натомість інші значення тримаються стабільно. Це можна пояснити тим, що кількість сторінок 400, а завдання можуть займати від 70 до 100 — достатньо велику частину пам'яті. Тому затримки і виникають. Для інших вихідних значень система працює стабільно, і всі відхилення менші 2%.
2	Система працює стабільно, і всі відхилення менші 2%.
3	
4	
5	
6	Бачимо, що відхилення часу на виділення пам'яті досягає 17%, час завдання в системі — 16.5%. Враховуючи те, що у порівнянні з набором 5 набір 6 відрізняється лише середнім інтервалом надходження завдання, і в наборі 6 він менший, то ці відхилення можна пояснити тим, що система перевантажена. Для інших вихідних значень система працює стабільно, і всі відхилення менші 2%.

3.3.3.2 Середньоквадратичні відхилення

Розраховані середньоквадратичні відхилення показали значні відносні коливання для завантажень дисків, каналу введення-виведення та процесорів, які досягають 42%. Водночас абсолютні значення цих відхилень є малими, що

робить їх дуже чутливими до незначних змін. Через це відносні відхилення для навантажень можна вважати несуттєвими. Для наборів 1 та 6 відносні відхилення в середніх квадратичних значеннях також є значними, з аналогічних причин, що й для середніх значень у таблиці 3.2.

3.4 Висновки до розділу

У цьому розділі було детально розглянуто алгоритм імітації мережі Петрі, що є основою для моделювання функціонування системи. На основі наявної бібліотеки PetriObjModelPaint було розроблено клас CourseWorkPetriSim, який реалізує основні етапи імітації, такі як просування часу, обробка переходів, вирішення конфліктів, збирання статистики та логування подій. Розроблений алгоритм був адаптований для задачі, що досліджується, із врахуванням специфічних вимог до роботи моделі.

Описані модифікації, внесені до початкової структури моделі, включають: додавання розподілу Пуассона, вдосконалення роботи з NetLibrary, покращення графічного інтерфейсу.

Було описано реалізацію CourseWorkNet для визначення структури мережі Петрі і основних методів імітації, що дозволяють запускати модель.

Було розглянуто клас AnalyzeCourseWorkNet, який використовується для збору статистичних даних моделі за різних вхідних параметрів.

Було описано також процес обробки зібраних даних для верифікації моделі. З цією метою було використано мову програмування Python із застосуванням бібліотек Pandas та NumPy. Для аналізу було розроблено спеціальні функції для розрахунку середніх значень та середньоквадратичних відхилень, що дозволяє оцінювати поведінку системи за кожним прогоном моделювання. Окрім розрахунків базових статистичних характеристик, було виконано обчислення глобальних середніх значень та середньоквадратичних відхилень для кожного набору параметрів. Крім того, було оцінено відсоткові відхилення кожного значення відносно глобального середнього, що дозволило

виділити нестандартну поведінку окремих прогонів. Результати обробки були представлені у вигляді таблиць.

Аналіз зібраних даних показав, що модель функціонує стабільно за більшістю наборів параметрів. Виявлені аномальні значення були пояснені особливостями параметризації, такими як перевантаження системи через занадто короткі інтервали між надходженням завдань. Верифікація також продемонструвала, що зміна вхідних параметрів моделі має очікуваний вплив на її поведінку, що підтверджує правильність реалізації алгоритму та структури моделі.

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МОДЕЛІ

У даному розділі будуть визначені відгуки: середні значення та середньоквадратичні відхилення вихідних параметрів моделі —, а також типи їхніх розподілів та перехідні періоди.

4.1 Визначення перехідного періоду

Перехідний період — це час, необхідний системі для досягнення стабільного стану, коли її поведінка починає відповідати реальним або усталеним умовам роботи. Перехідний період у моделюванні необхідно визначати для забезпечення коректності та точності результатів. У процесі імітації моделі системи, зокрема мережі Петрі, перші моменти моделювання не відображають стабільний стан системи. Це пов'язано з тим, що система лише починає працювати, і її стан змінюється під впливом початкових умов.

Для початку завантажимо дані та відсортуємо за часом:

```

1 from pathlib import Path
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 folder_path = Path('final_data')
6 common_props_raw: pd.DataFrame = pd.read_csv(folder_path / 'commonProps.csv')
7 time_in_system_raw: pd.DataFrame = pd.read_csv(folder_path / 'timeInSystem.csv')
8 time_wait_allocate_raw: pd.DataFrame = pd.read_csv(folder_path / 'timeWaitAllocate.csv')
9 common_props_raw.sort_values(by='timePoint', inplace=True)
10 time_in_system_raw.sort_values(by='timePoint', inplace=True)
11 time_wait_allocate_raw.sort_values(by='timePoint', inplace=True)

```

4.1.1 Параметри за замовчуванням

Спробуємо визначити перехідні періоди для параметрів, що визначенні в умові завдання, набір 6 з таблиці А.1. Для цього аналогічно до пункту 3.3.1 проженемо модель 5 разів за однакових параметрів.

4.1.1.1 Вихідні параметри в момент часу

Після збору даних потрібно визначити середні значення та середньоквадратичні відхилення в кожен момент часу моделювання. Щоб це зробити були визначені функції `calculate_means_through_time` та `calculate_stddevs_through_time`:

```

1 import numpy as np
2 from array import array
3 from typing import Sequence
4 def calculate_means_through_time(
5     time_points: pd.Series,
6     values: pd.Series
7 ) -> array[float]:
8     prev_time_point = time_points.iloc[0]
9     delay_sum = 0.0
10    value_sum = 0.0
11    values_through_time = array('d')
12    for time_point, value in zip(time_points.iloc[1:], values):
13        delay = time_point - prev_time_point
14        prev_time_point = time_point
15        delay_sum += delay
16        value_sum += value * delay
17        values_through_time.append(value_sum / delay_sum)
18    return values_through_time
19 def calculate_stddevs_through_time(
20     time_points: pd.Series,
21     values: pd.Series,
22     means: array[float]
23 ) -> array[float]:
24     prev_time_point = time_points.iloc[0]
25     delay_sum = 0.0
26     value_sum = 0.0
27     stddevs_through_time: array[float] = array('d')
28     for time_point, value, mean in zip(time_points.iloc[1:], values, means):

```

```

29     delay = time_point - prev_time_point
30     prev_time_point = time_point
31     delay_sum += delay
32     value_sum += ((value - mean) ** 2) * delay
33     stddevs_through_time.append(np.sqrt(value_sum / delay_sum))
34     return stddevs_through_time

```

Далі обчислимо параметри:

```

1 from collections import deque
2 import attr
3
4 @attr.frozen
5 class PropertyMeanStdDev:
6     mean: array[float]
7     stdDev: array[float]
8
9 def calc_mean_stddev_through_time(time_points: pd.Series, props: pd.Series) ->
PropertyMeanStdDev:
10     means = calculate_means_through_time(time_points, props)
11     return PropertyMeanStdDev(
12         means,
13         calculate_stddevs_through_time(time_points, props, means)
14     )
15
16 time_points_mat: deque[Sequence[float]] = deque()
17 disk_load_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
18 io_channel_load_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
19 processors_load_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
20 use_of_page_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
21 total_wait_allocate_stddev_mat: deque[PropertyMeanStdDev] = deque()
22
23 time_in_system_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
24 time_wait_allocate_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
25 time_in_system_time_points_mat: deque[PropertyMeanStdDev] = deque()
26 time_wait_allocate_time_points_mat: deque[PropertyMeanStdDev] = deque()

```

```

27
28 for run_number, common_props_raw_indexed in
common_props_raw.groupby(by='runNumber'):
29     time_points = common_props_raw_indexed['timePoint'][:-1]
30     disk_load_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['diskLoad']))
31     io_channel_load_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['ioChannelLoad']))
32     processors_load_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['processorsLoad']))
33     use_of_page_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['useOfPage']))
34     total_wait_allocate_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['totalWaitAllocate']))
35     time_points_mat.append(time_points[:-1])
36
37 for run_number, time_in_system_raw_indexed in
time_in_system_raw.groupby(by='runNumber'):
38     time_in_system_time_points = time_in_system_raw_indexed['timePoint']
39
time_in_system_mean_stddev_mat.append(calc_mean_stddev_through_time(time_in_system_time
_points, time_in_system_raw_indexed['timeInSystem']))
40     time_in_system_time_points_mat.append(time_in_system_time_points.iloc[:-1])
41
42 for run_number, time_wait_allocate_raw_indexed in
time_wait_allocate_raw.groupby(by='runNumber'):
43     time_wait_allocate_time_points = time_wait_allocate_raw_indexed['timePoint']
44
time_wait_allocate_mean_stddev_mat.append(calc_mean_stddev_through_time(time_wait_alloca
te_time_points, time_wait_allocate_raw_indexed['timeWaitAllocate']))
45     time_wait_allocate_time_points_mat.append(time_wait_allocate_time_points.iloc[:-1])

```

Далі побудуємо графіки залежності відгуків від часу моделювання на
 рисунках 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14:

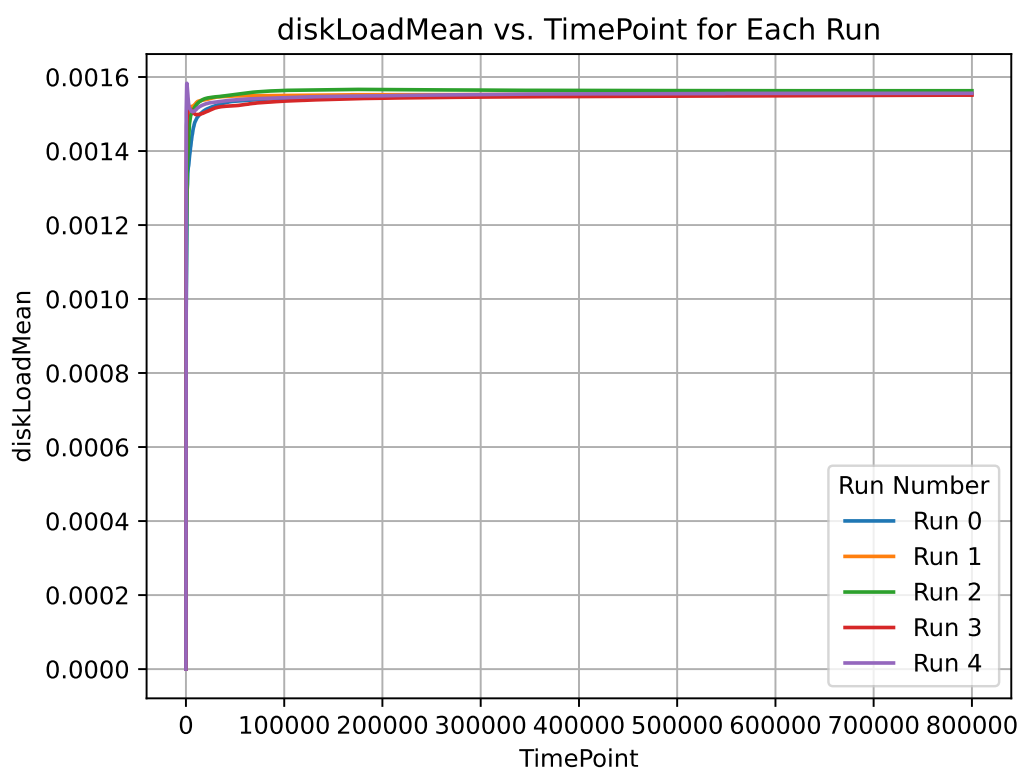


Рисунок 4.1 — Залежність середнього значення навантаження диска від часу

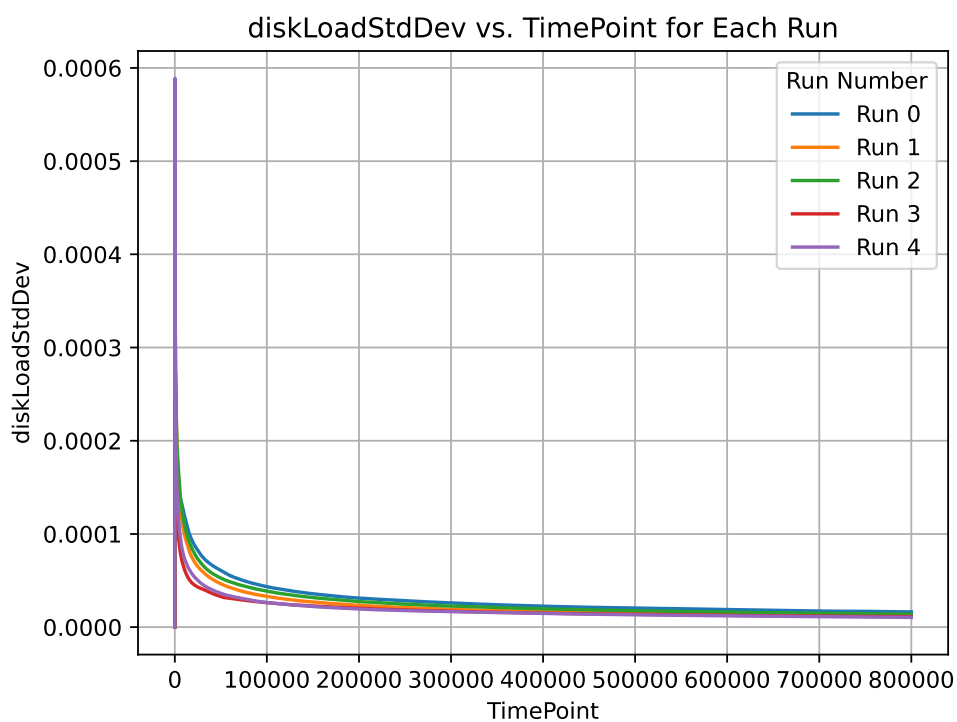


Рисунок 4.2 — Залежність середньоквадратичного відхилення навантаження диска від часу

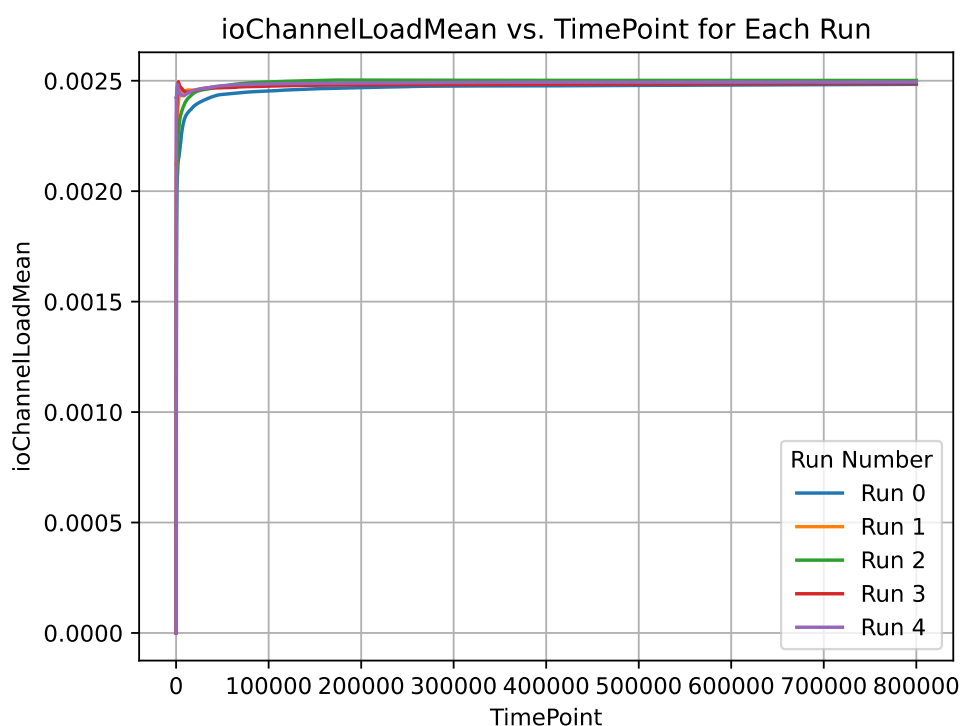


Рисунок 4.3 — Залежність середнього значення навантаження каналу передачі від часу

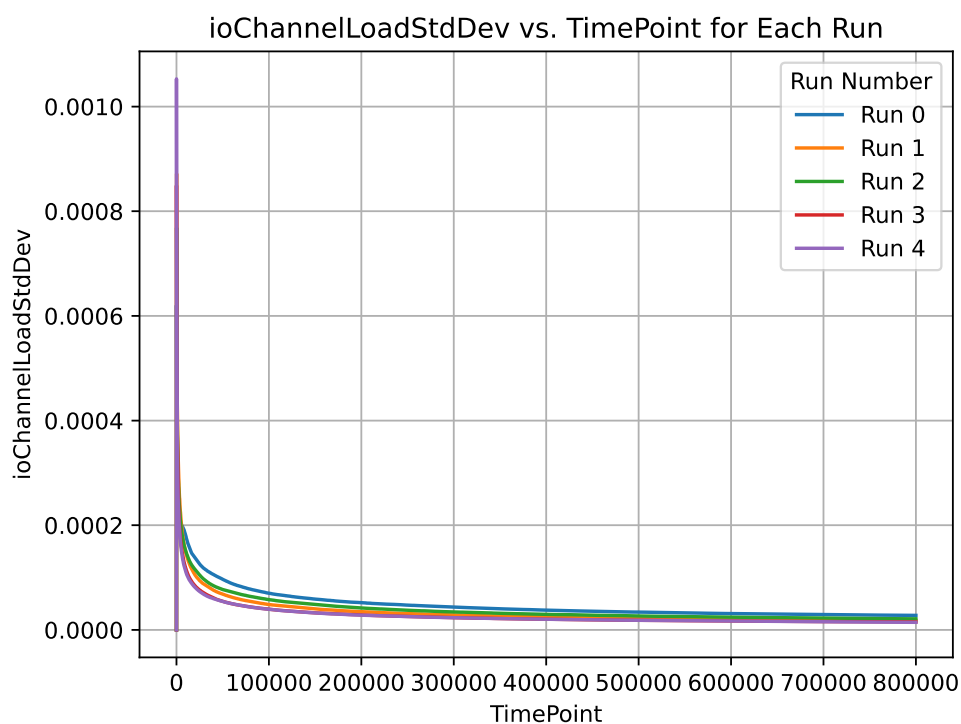


Рисунок 4.4 — Залежність середньоквадратичного відхилення навантаження каналу передачі від часу

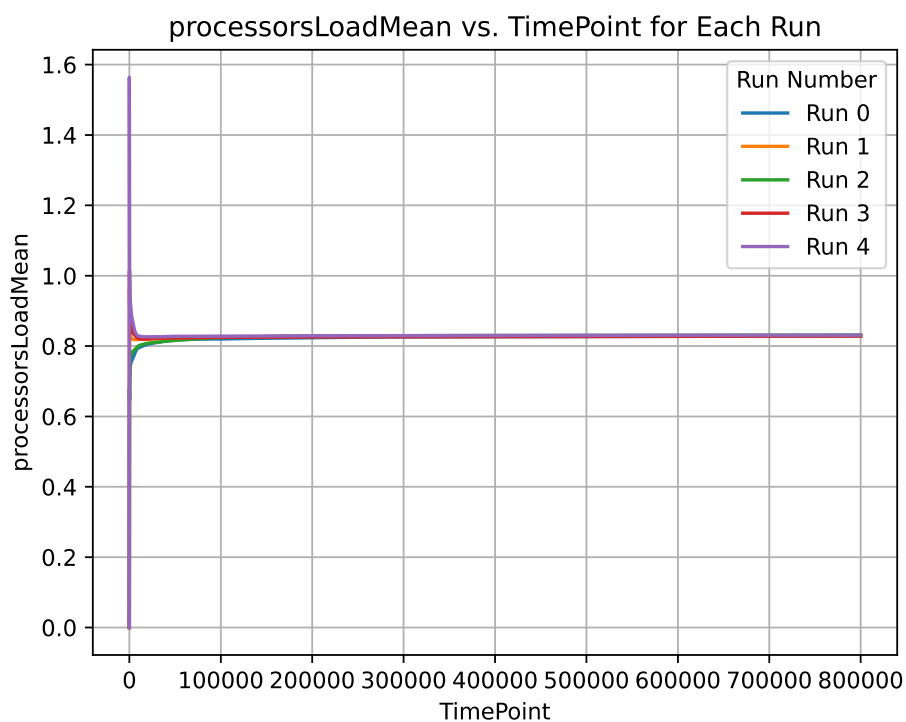


Рисунок 4.5 — Залежність середнього значення навантаження процесора від часу

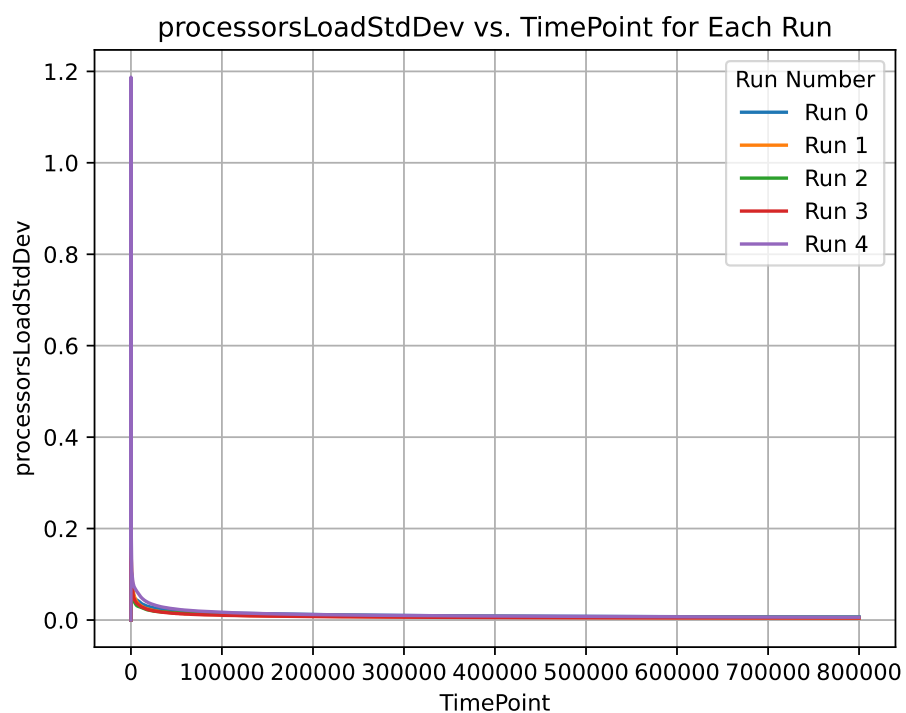


Рисунок 4.6 — Залежність середньоквадратичного відхилення навантаження процесора від часу

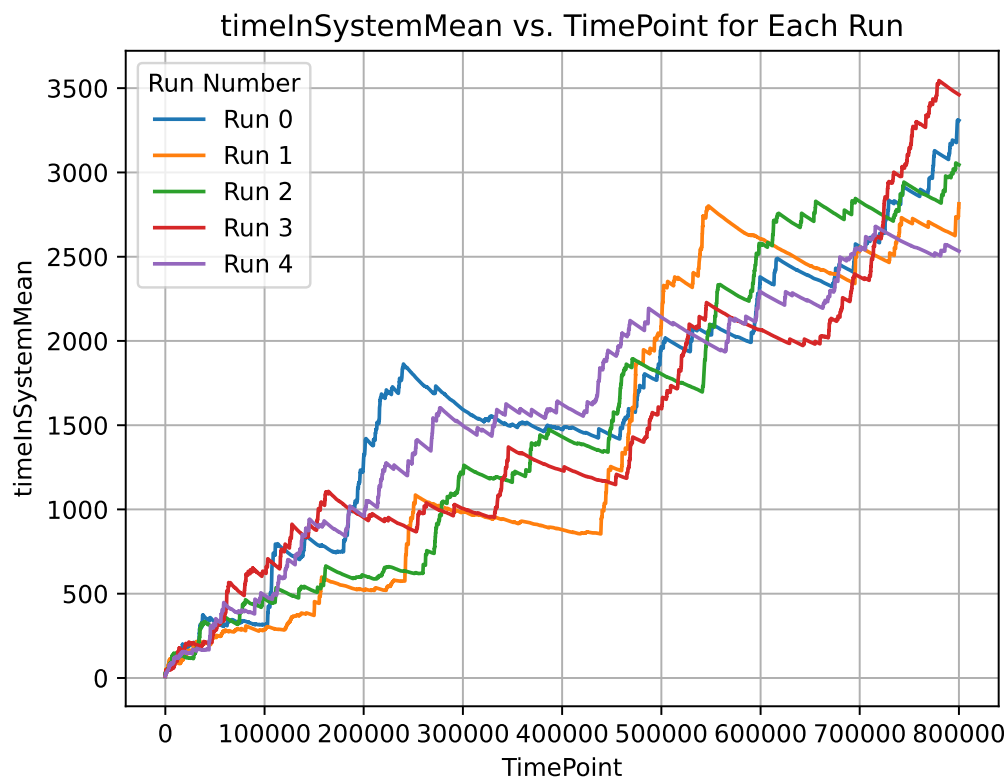


Рисунок 4.7 — Залежність середнього часу завдання в системі від часу

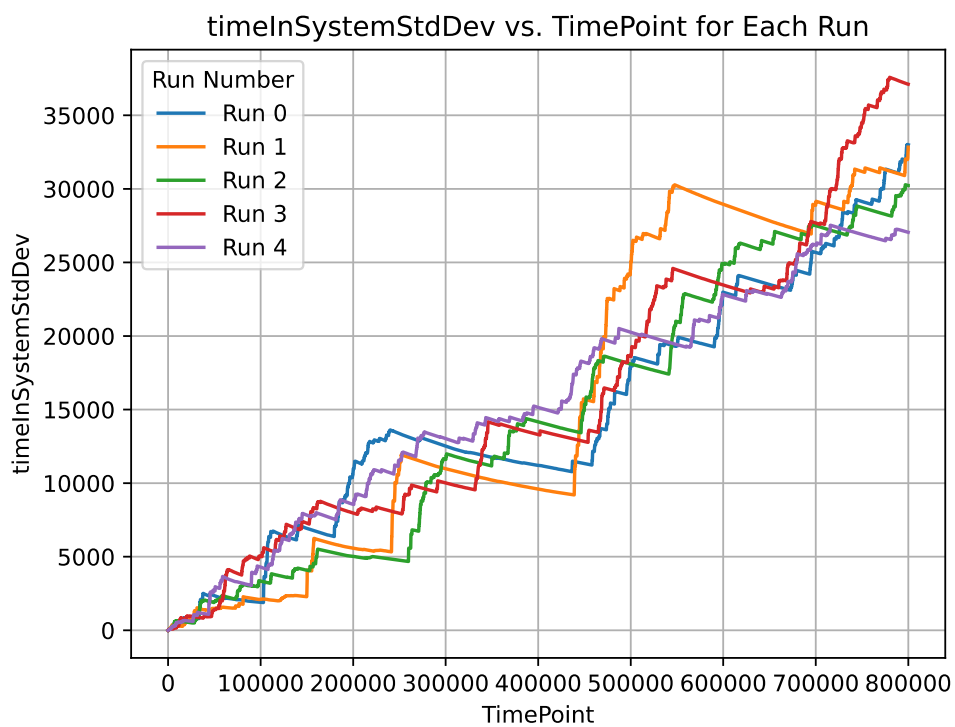


Рисунок 4.8 — Залежність середньоквадратичного відхилення часу завдання в системі від часу

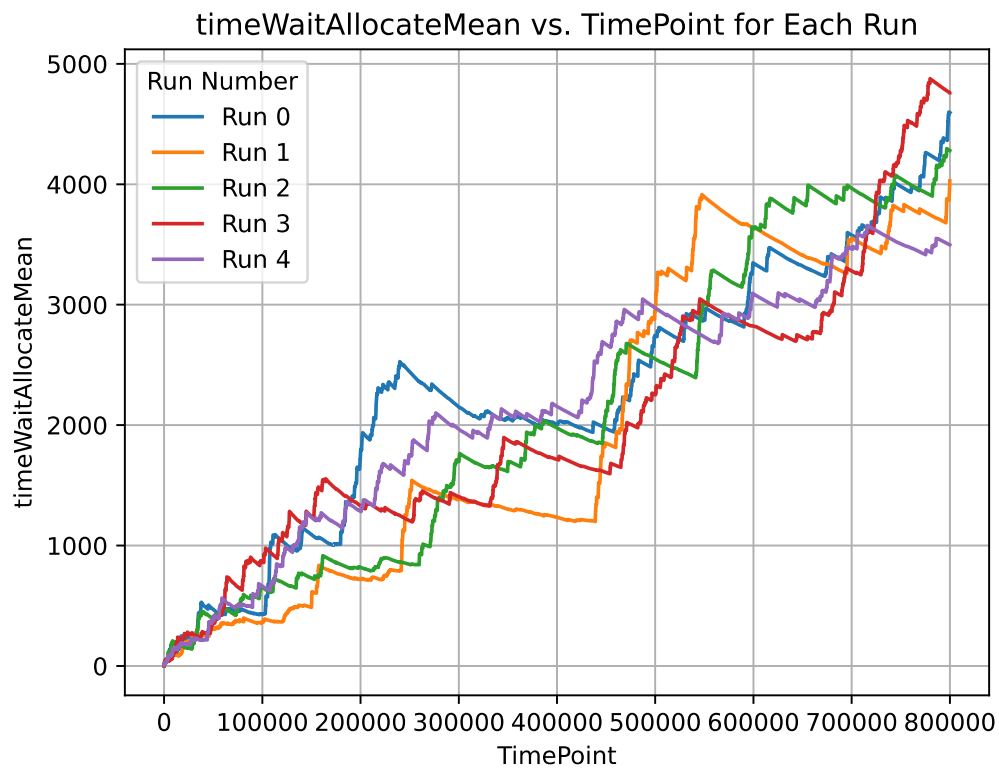


Рисунок 4.9 — Залежність середнього значення часу очікування пам'яті від часу

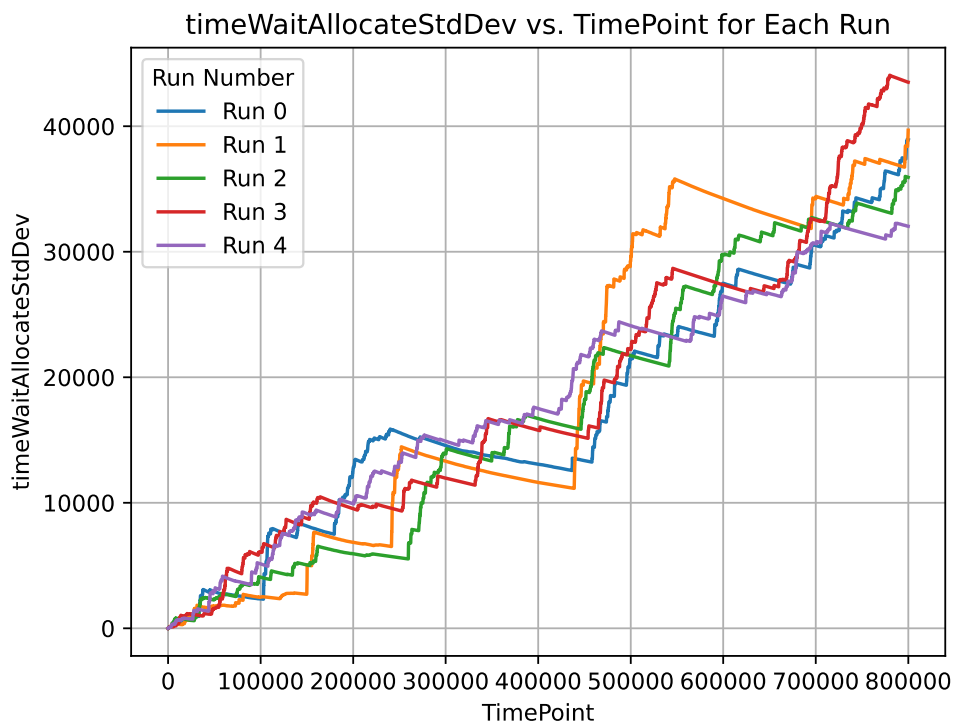


Рисунок 4.10 — Залежність середньоквадратичного відхилення часу очікування пам'яті від часу

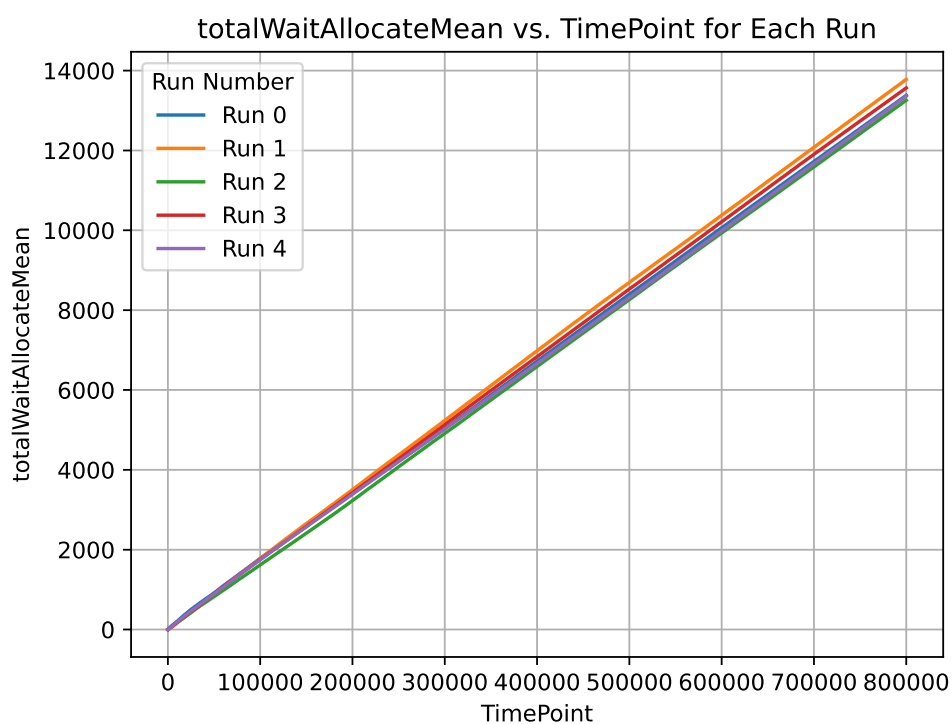


Рисунок 4.11 — Залежність середнього значення кількості завдань, що очікують виділення пам'яті, від часу

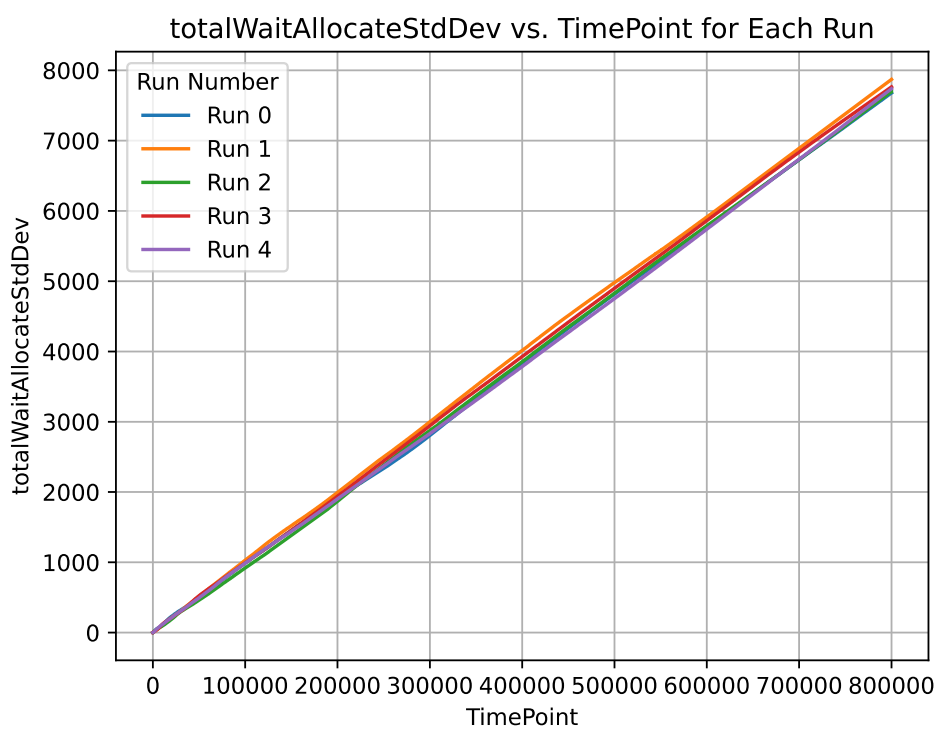


Рисунок 4.12 — Залежність середньоквадратичного відхилення кількості завдань, що очікують виділення пам'яті, від часу

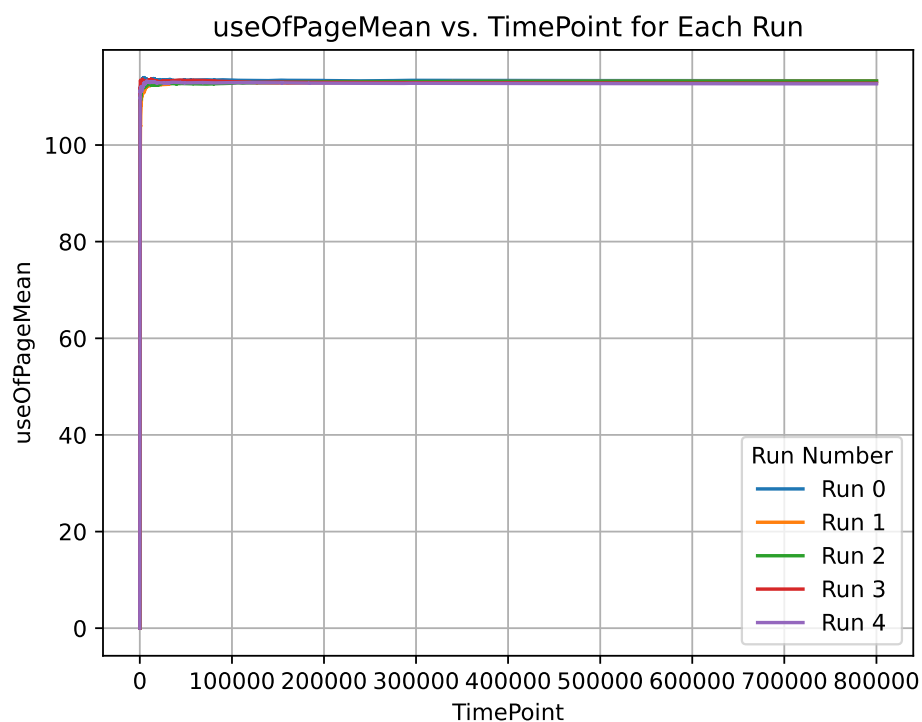


Рисунок 4.13 — Залежність середнього значення кількості зайнятих сторінок від часу

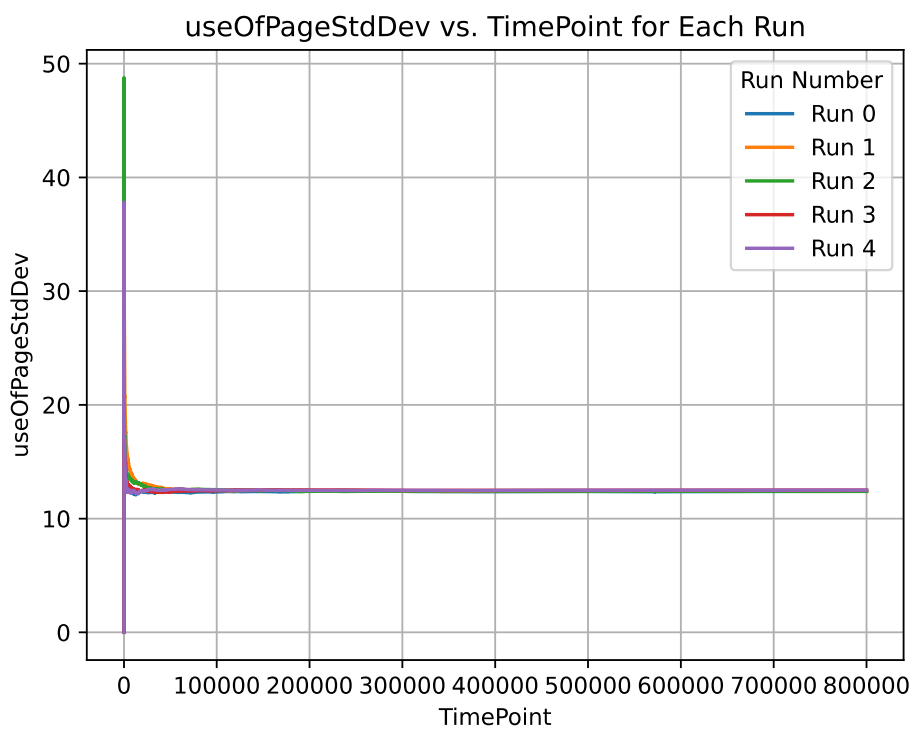


Рисунок 4.14 — Залежність середньоквадратичного відхилення кількості зайнятих сторінок від часу

З рисунків 4.7, 4.8, 4.9, 4.10, 4.11, 4.12 бачимо, що система не може досягти сталих значень, що виникає з причин, описаних у таблиці 3.2.

4.1.2 Змінені параметри

Застосуємо параметри з набору 5 з таблиці A.1 на рисунках 4.15, 4.16, 4.17, 4.18, 4.19, 4.20, 4.21, 4.21, 4.22, 4.23, 4.24, 4.25, 4.26, 4.27, 4.28.

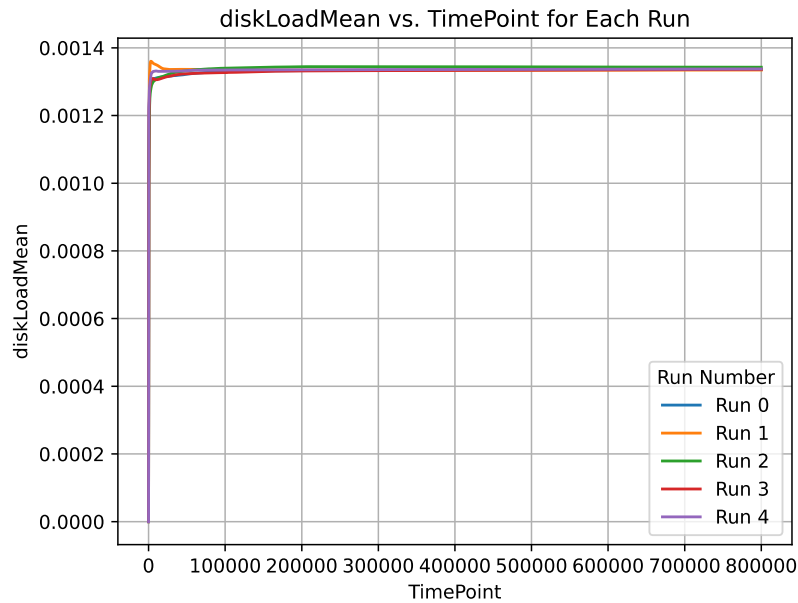


Рисунок 4.15 — Залежність середнього значення навантаження диска від часу

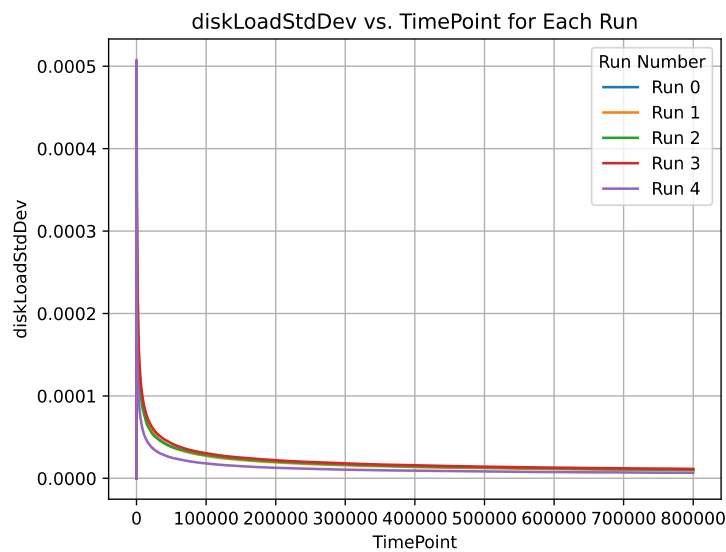


Рисунок 4.16 — Залежність середньоквадратичного відхилення навантаження диска від часу

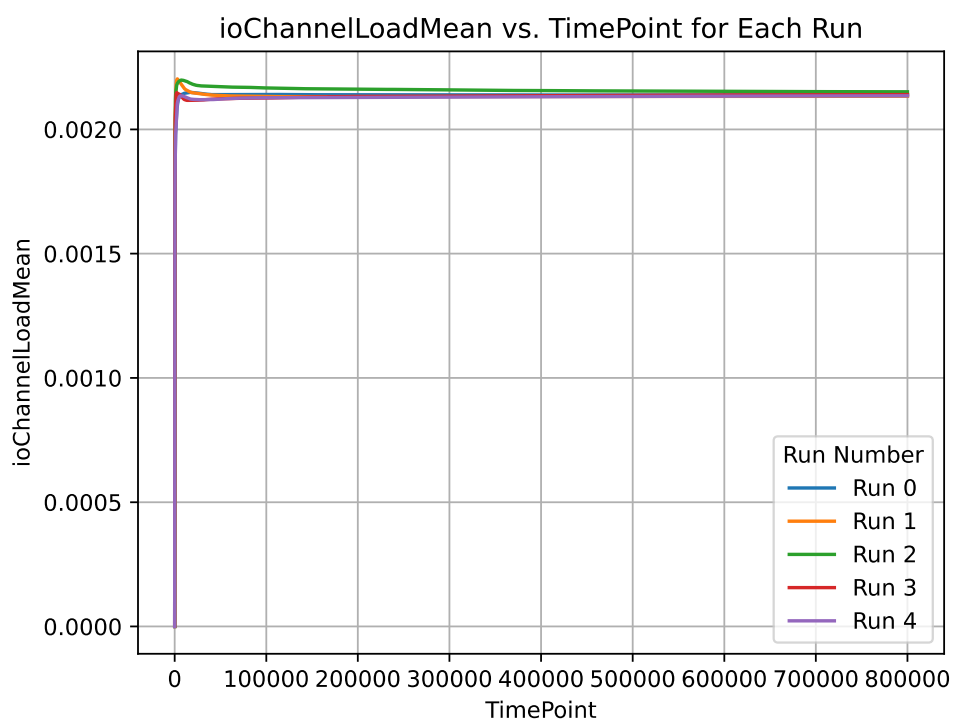


Рисунок 4.17 — Залежність середнього значення навантаження каналу передачі від часу

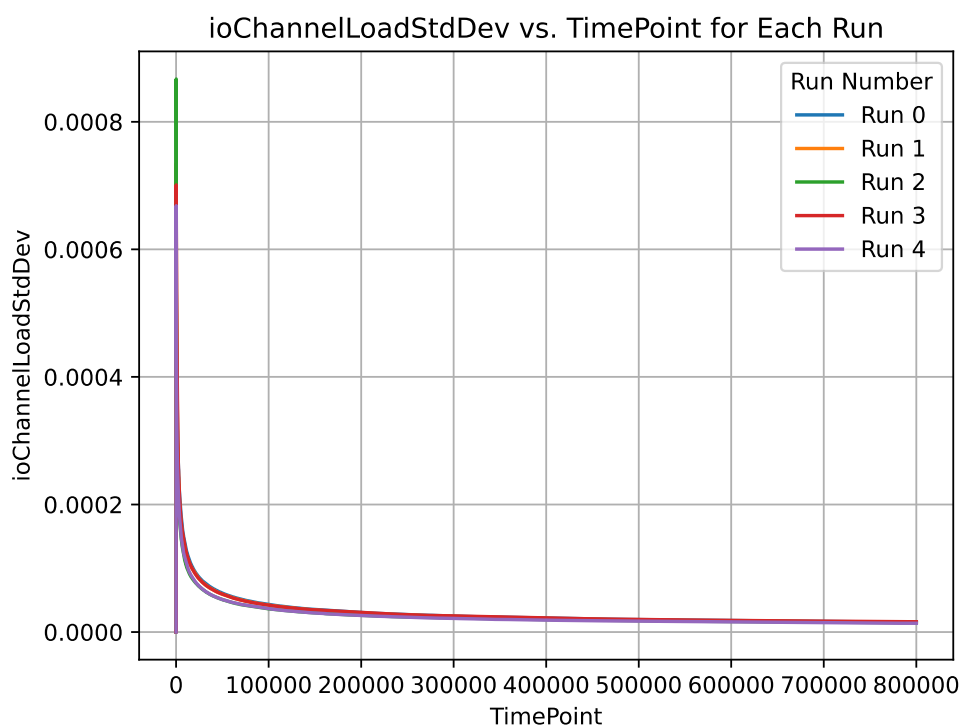


Рисунок 4.18 — Залежність середньоквадратичного відхилення навантаження каналу передачі від часу

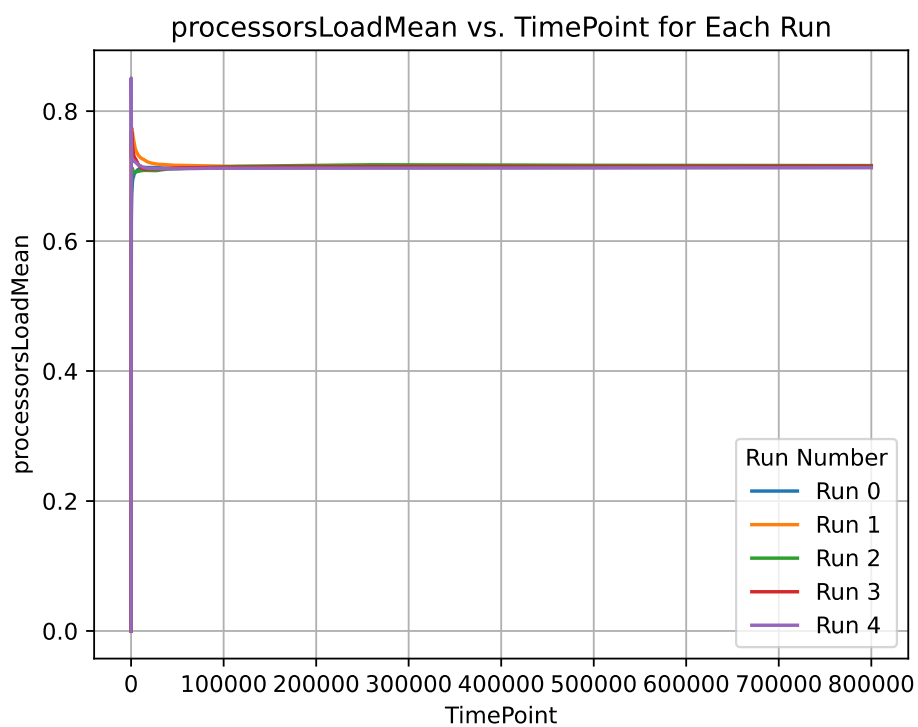


Рисунок 4.19 — Залежність середнього значення навантаження процесора від часу

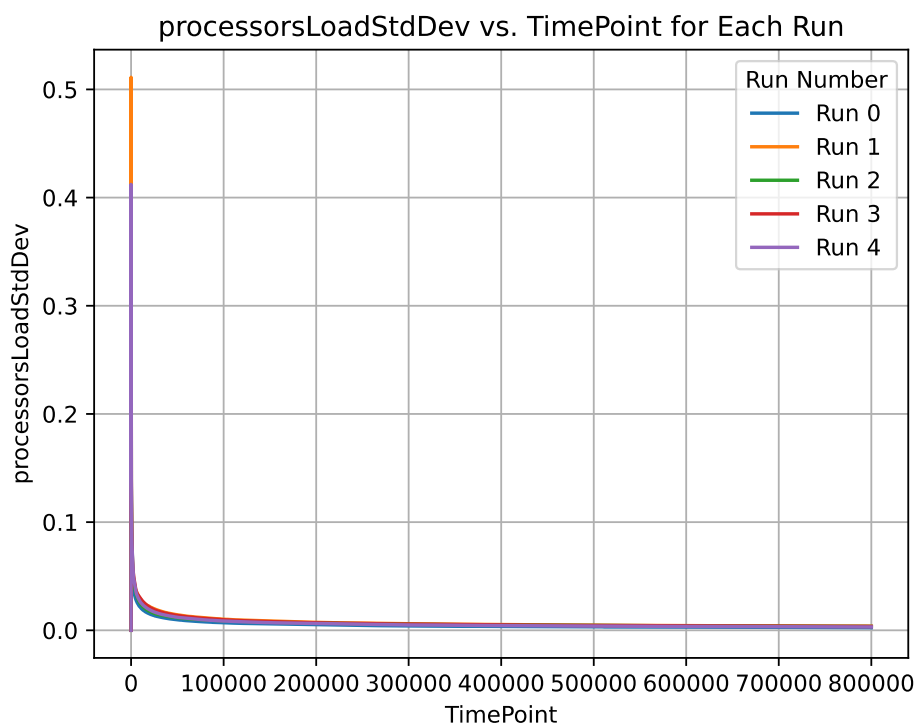


Рисунок 4.20 — Залежність середньоквадратичного відхилення навантаження процесора від часу

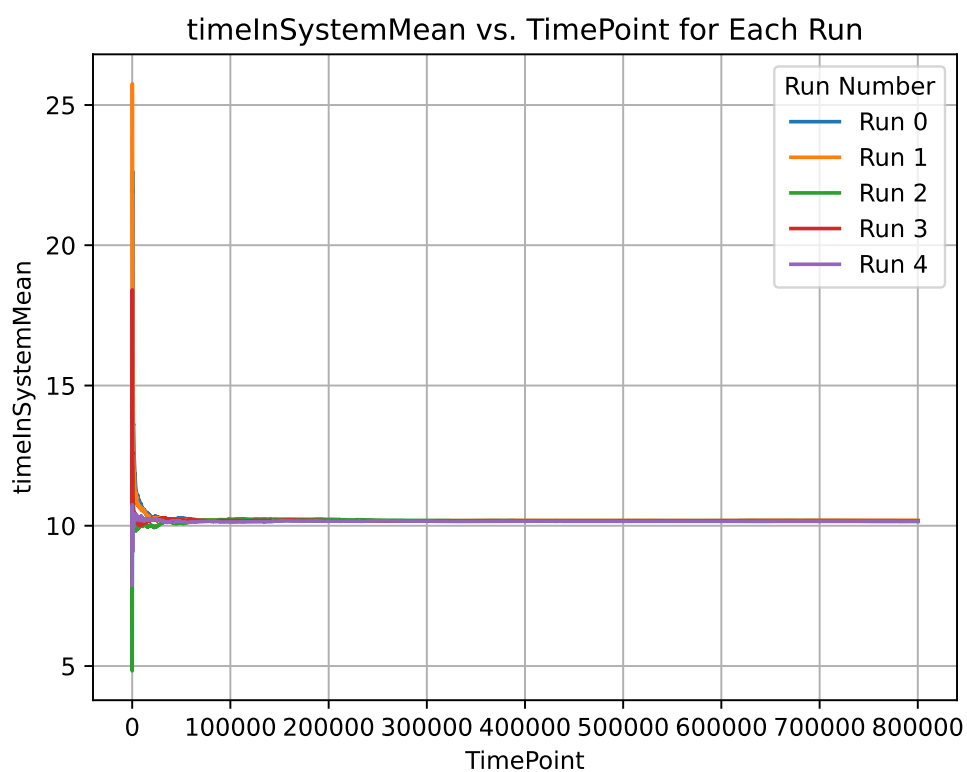


Рисунок 4.21 — Залежність середнього часу завдання в системі від часу

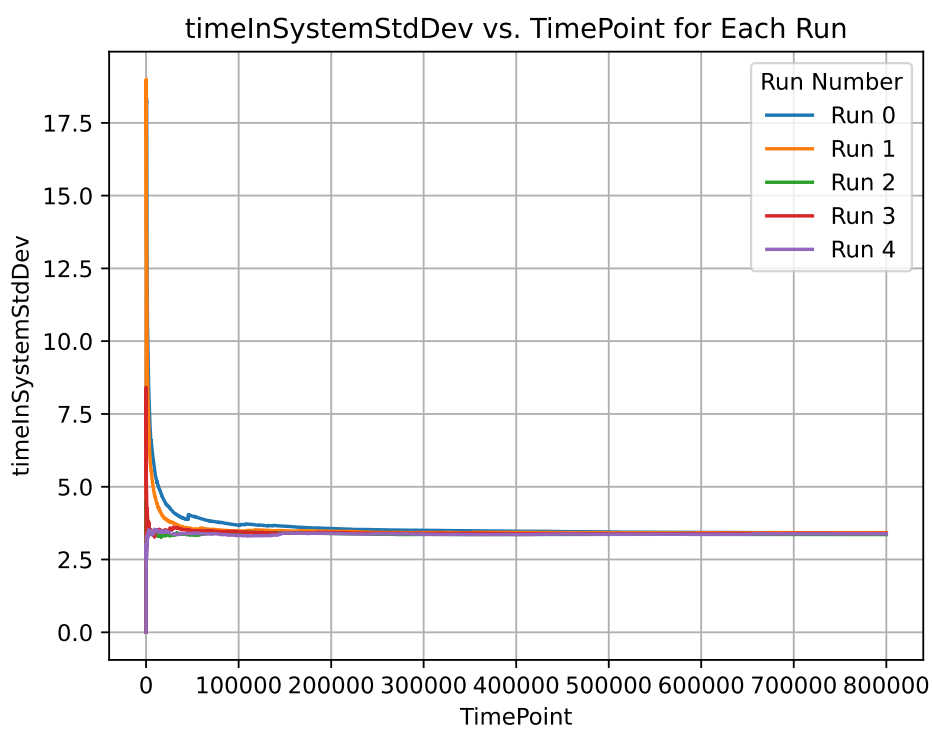


Рисунок 4.22 — Залежність середньоквадратичного відхилення часу завдання в системі від часу

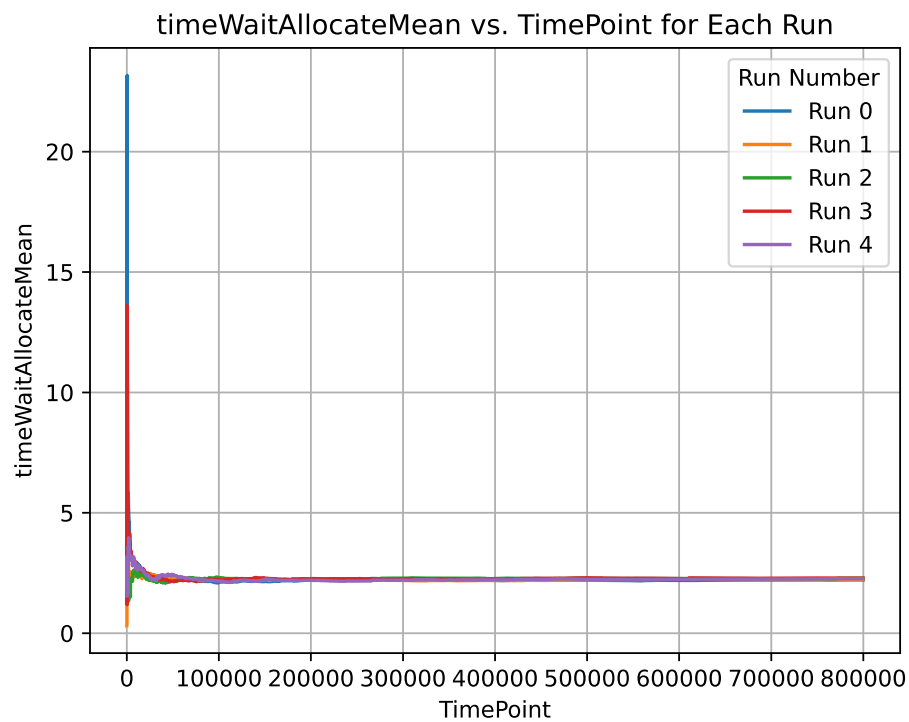


Рисунок 4.23 — Залежність середнього значення часу очікування пам'яті від часу

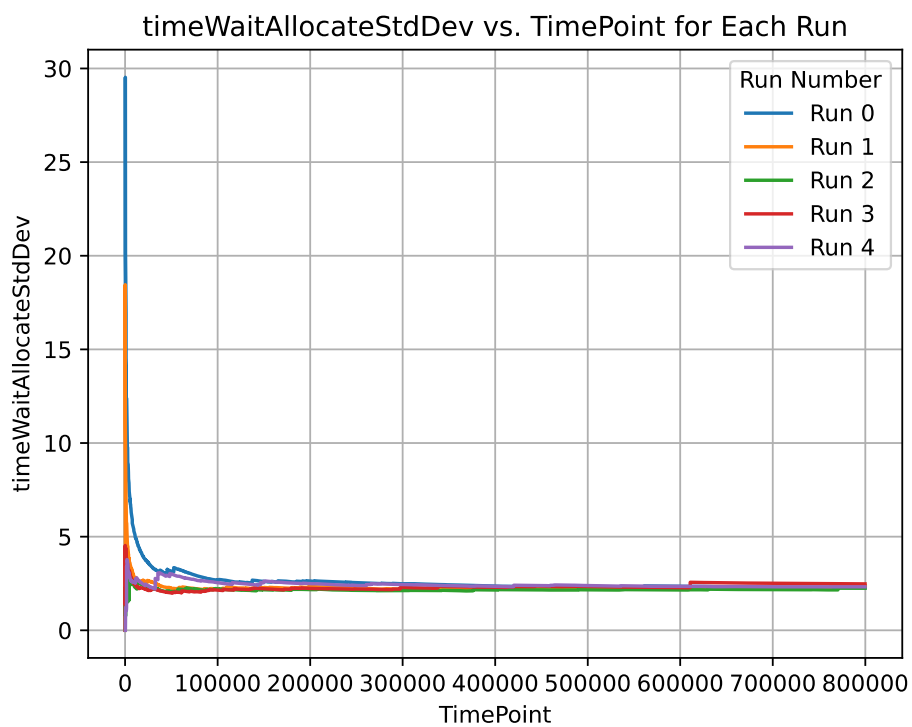


Рисунок 4.24 — Залежність середньоквадратичного відхилення часу очікування пам'яті від часу

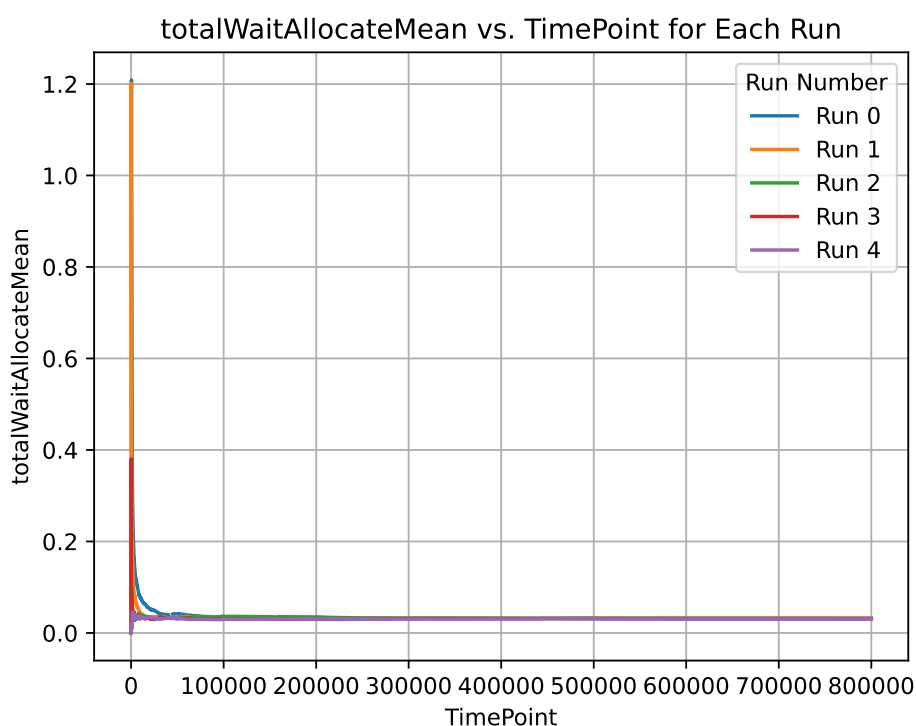


Рисунок 4.25 — Залежність середнього значення кількості завдань, що очікують виділення пам'яті, від часу

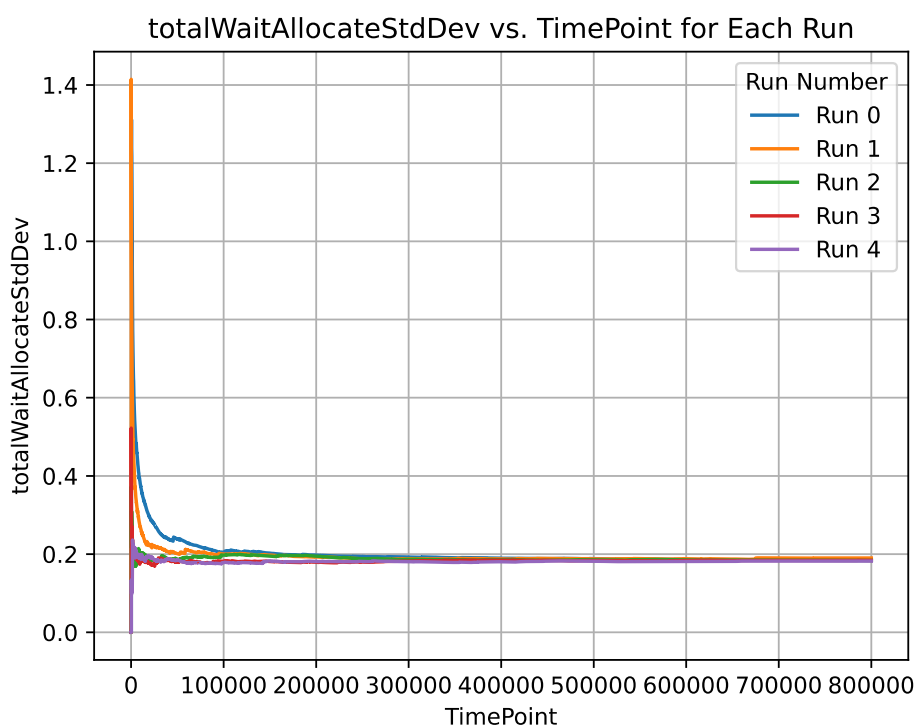


Рисунок 4.26 — Залежність середньоквадратичного відхилення кількості завдань, що очікують виділення пам'яті, від часу

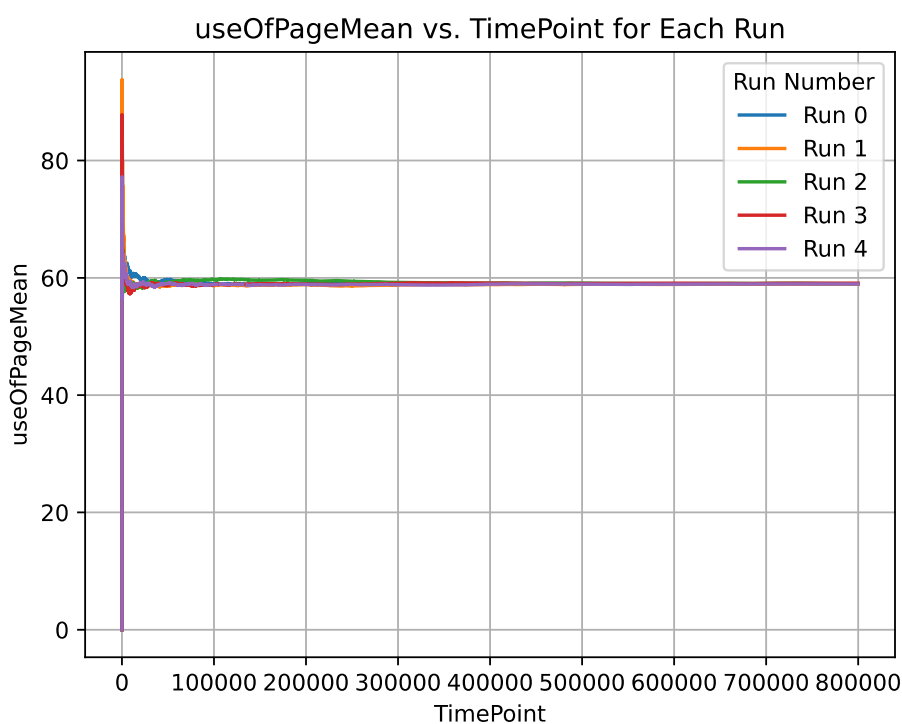


Рисунок 4.27 — Залежність середнього значення кількості зайнятих сторінок від часу

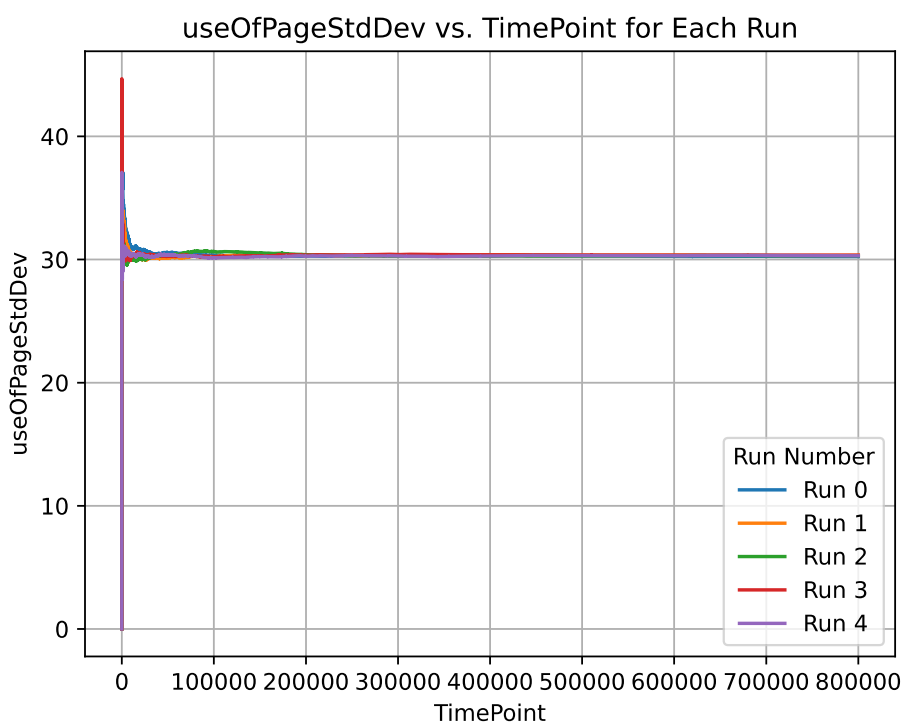


Рисунок 4.28 — Залежність середньоквадратичного відхилення кількості зайнятих сторінок від часу

Бачимо, що для того, щоб всі відгуки стабілізувалися, потрібно 600000 одиниць часу (далі $T_{\text{перехідний}}$).

4.2 Визначення кількості необхідних прогонів

Кількість замірів визначається за формулою (4.1), що є наслідком нерівності Чебишева:

$$p = \frac{\sigma^2}{\epsilon^2(1-\beta)} \quad (4.1)$$

де σ — середньоквадратичне відхилення величини, що спостерігається;

ϵ — точність, або бажане середньоквадратичне відхилення;

β — довірна ймовірність оцінювання.

Оскільки після моменту часу $T_{\text{перехідний}}$ відгуки на графіках зійшлися до константних значень, то достатньо, що $\epsilon = \sigma$. За умови що $\beta = 0.95$ за розрахунками (4.2) визначимо p :

$$p = \frac{\sigma^2}{\sigma^2(1-\beta)} = \frac{1}{0.05} = 20 \quad (4.2)$$

4.3 Визначення середніх значень та середньоквадратичних відхилень

У таблиці 4.1 визначено середні значення та середньоквадратичні відхилення:

```

1 global_mean: dict[str, float] = {}
2 global_stddev: dict[str, float] = {}
3
4 def update_global_mean_stddev(
5     dest_mean: dict[str, float],
6     dest_stddev: dict[str, float],
7     src: pd.DataFrame,
8     col: str
9 ) -> None:
10     mean = calculate_mean(src['timePoint'], src[col])
11     dest_mean[col] = mean

```

```

12  dest_stddev[col] = calculate_std_dev(src['timePoint'], src[col], mean)
13
14  for run_number, stable_run in common_props_stable.groupby('runNumber'):
15      update_global_mean_stddev(global_mean, global_stddev, stable_run, 'diskLoad')
16      update_global_mean_stddev(global_mean, global_stddev, stable_run, 'ioChannelLoad')
17      update_global_mean_stddev(global_mean, global_stddev, stable_run, 'processorsLoad')
18      update_global_mean_stddev(global_mean, global_stddev, stable_run, 'totalWaitAllocate')
19      update_global_mean_stddev(global_mean, global_stddev, stable_run, 'useOfPage')
20
21  for run_number, stable_run in time_in_system_stable.groupby('runNumber'):
22      update_global_mean_stddev(global_mean, global_stddev, stable_run, 'timeInSystem')
23
24  for run_number, stable_run in time_wait_allocate_stable.groupby('runNumber'):
25      update_global_mean_stddev(global_mean, global_stddev, stable_run, 'timeWaitAllocate')

```

Таблиця 4.1 Середні значення та середньоквадратичні відхилення вихідних параметрів

Назва вихідного параметра	Середнє значення	Середньоквадратичне відхилення
Завантаження дисків	0.0013377687709915573	5.164305642651139e-07
Завантаження каналу введення-виведення	0.0021408845049947366	5.826438943862089e-07
Завантаження процесорів	0.7136420069832722	0.0002284169580247951
Кількість завдань в очікуванні пам'яті	0.021118337643089388	0.1499305819609746
Кількість зайнятих сторінок	55.119726204499884	30.542913857411254
Час завдання в системі	10.103186785929791	3.308639012833178
Час виділення пам'яті	2.1739841157524746	2.0189247170561306

4.4 Визначенні типів розподілів

На рисунках 4.29, 4.30, 4.31, 4.32, 4.33, 4.34, 4.35 зобразимо гістограми розподілів.

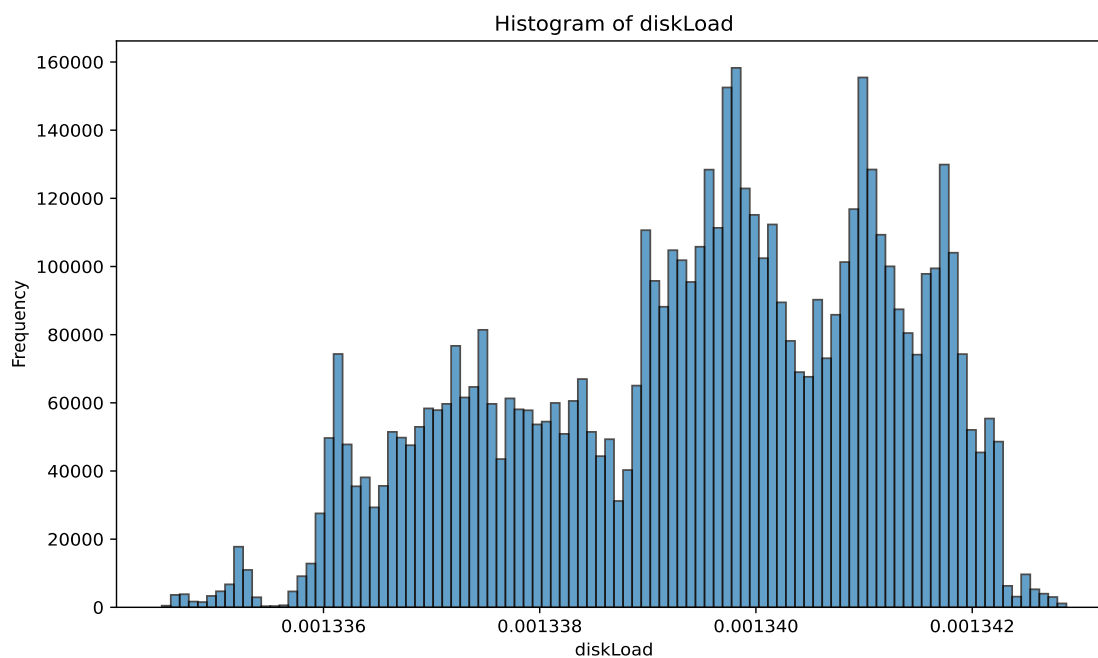


Рисунок 4.29 — Гістограма розподілу величини завантаження диска

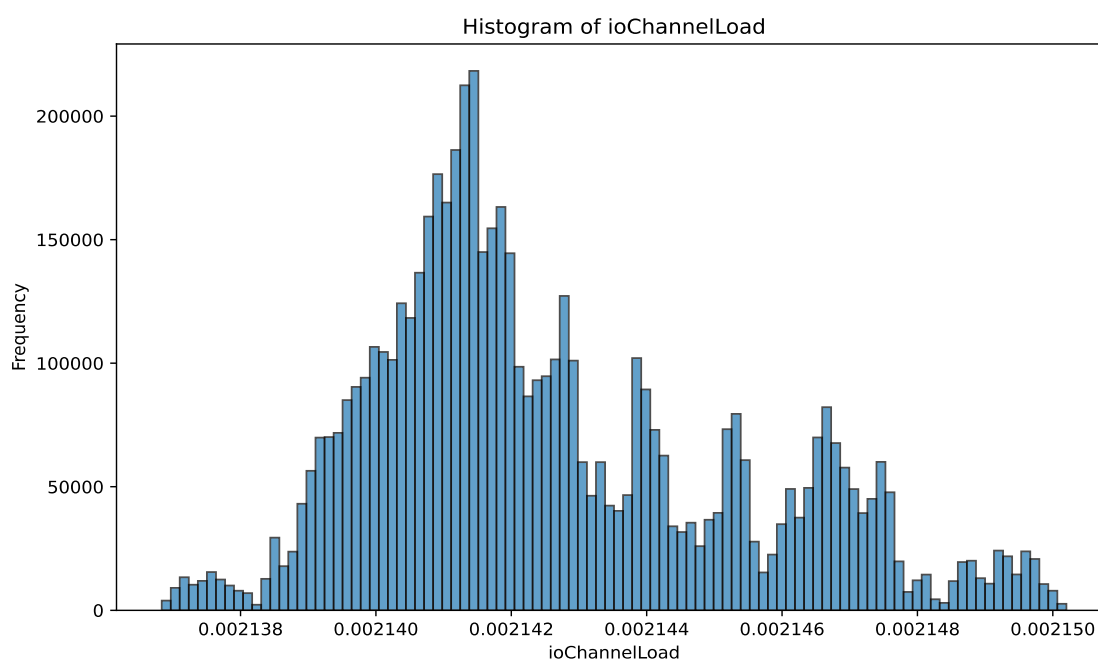


Рисунок 4.30 — Гістограма розподілу навантаження каналу передачі

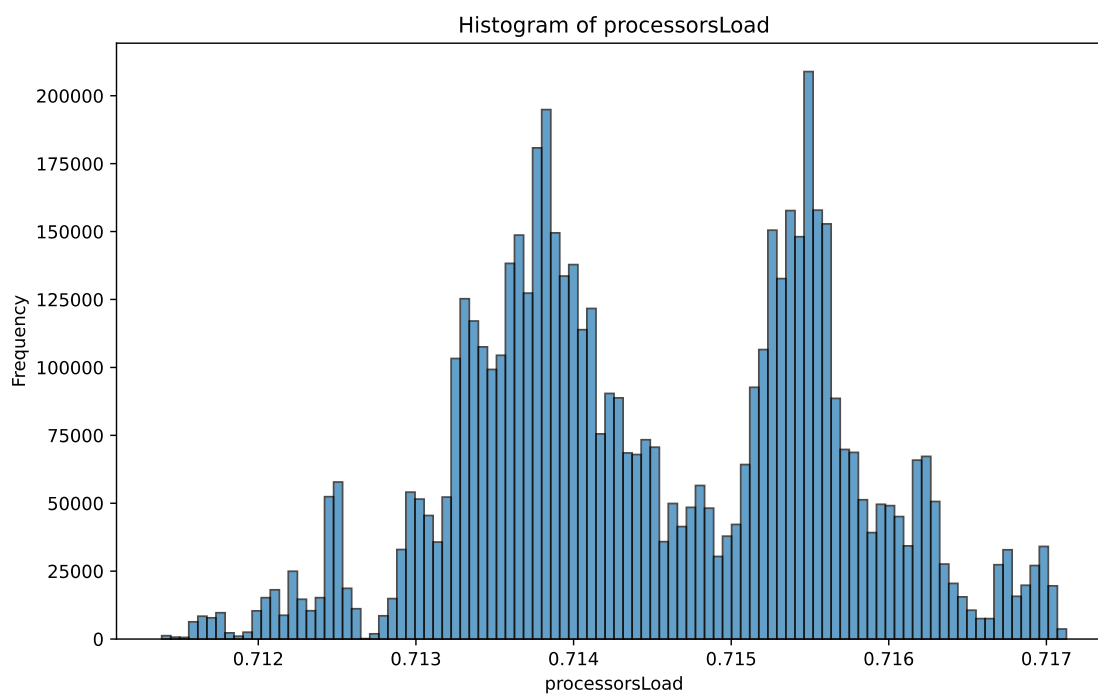


Рисунок 4.31 — Гістограма розподілу навантаження процесора

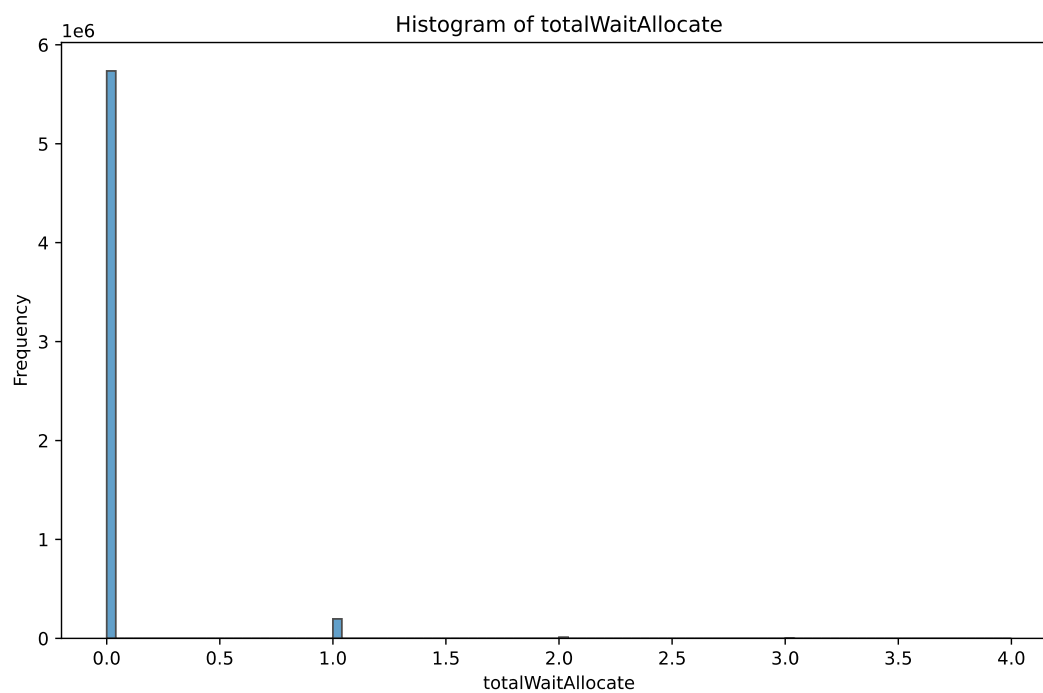


Рисунок 4.32 — Гістограма розподілу кількості завдань, що очікують виділення пам'яті

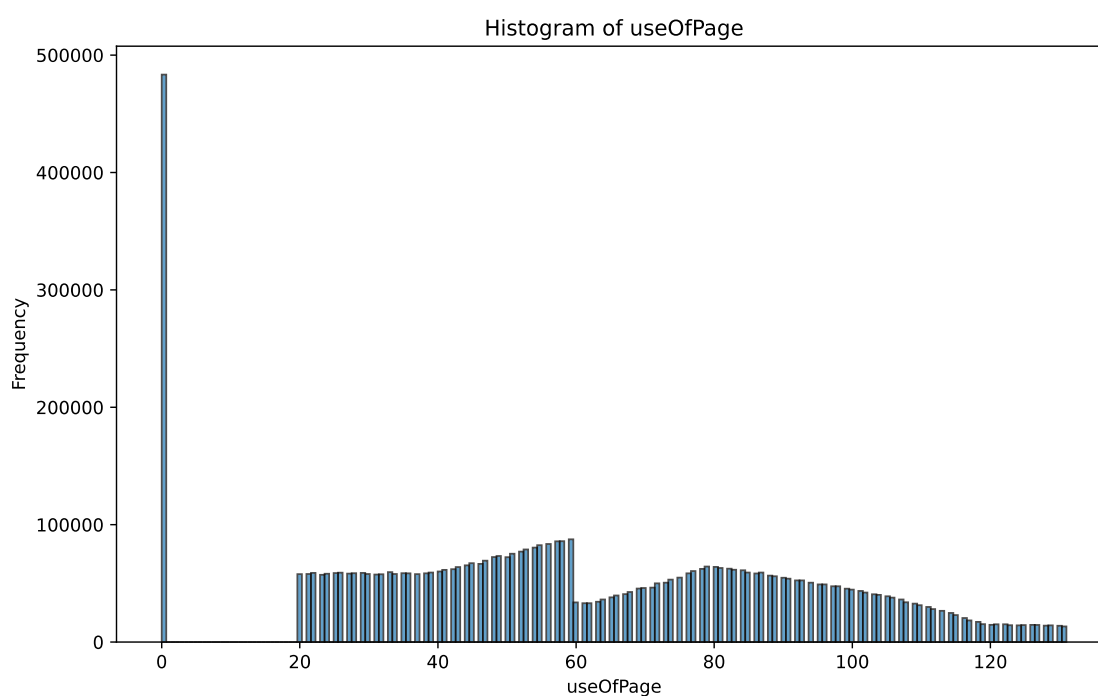


Рисунок 4.33 — Гістограма розподілу кількості зайнятих сторінок пам'яті

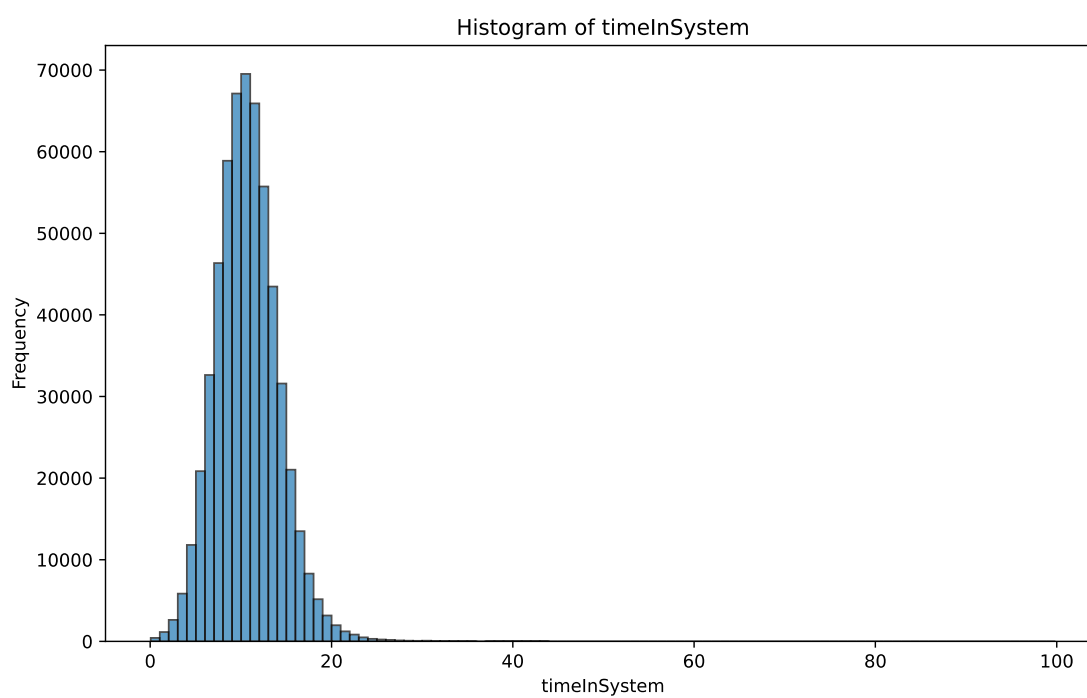


Рисунок 4.34 — Гістограма розподілу кількості часу завдання в системі

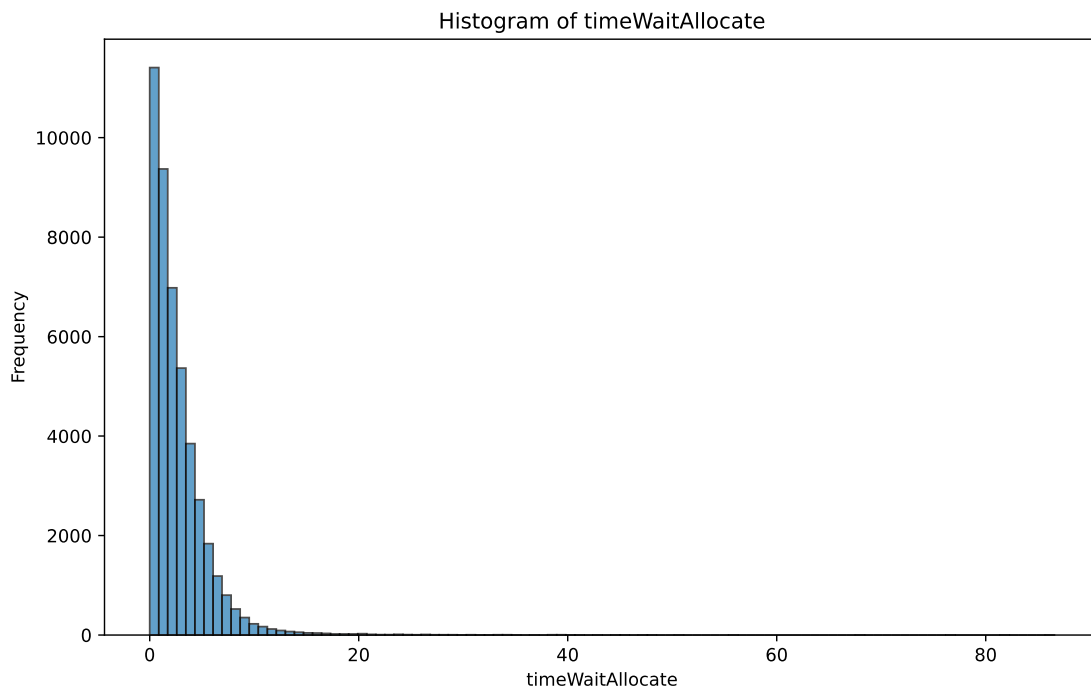


Рисунок 4.35 — Гістограма розподілу кількості часу, що очікує завдання на виділення пам'яті

4.4.1 Аналіз розподілів навантаження диска, процесора, каналу передачі

З таблиці 4.1 бачимо, що відхилення дуже близькі до 0. Тому можна сказати, що навантаження сходяться до постійних значень. На рисунках 4.29, 4.30, 4.31 ми бачимо лише надзвичайно малі в абсолютних значеннях коливання, які не піддаються певному визначеному розподілу.

4.4.2 Аналіз розподілу кількості завдань в очікуванні пам'яті

Бачимо, що майже увесь час роботи системи немає завдань, що очікують виділення пам'яті, а тому розподілом є постійне значення.

4.4.3 Аналіз розподілу кількості часу завдання в системі

Бачимо, що розподіл схожий на нормальний, перевіримо цю гіпотезу критерієм згоди χ^2 , для цього визначимо функцію `chi_squared_normality_test`:

```
1 def chi_squared_normality_test(data: Sequence[float], mean: float, std_dev: float) -> bool:
```

```
2  # Розрахунок кількості бінів за правилом Стерджеса
3  num_bins = int(np.ceil(1 + 3.322 * np.log10(len(data))))
4
5  # Створення гістограми даних
6  observed_counts, bin_edges = np.histogram(data, bins=num_bins)
7
8  # Розрахунок очікуваних частот для нормального розподілу
9  expected_counts = []
10 for i in range(len(bin_edges) - 1):
11     # Розрахунок ймовірності для кожного інтервалу
12     bin_prob = (
13         (1 / (std_dev * np.sqrt(2 * np.pi))) *
14         (np.exp(-0.5 * ((bin_edges[i + 1] - mean) / std_dev) ** 2) -
15         np.exp(-0.5 * ((bin_edges[i] - mean) / std_dev) ** 2))
16     )
17     # Додавання очікуваної частоти для даного інтервалу
18     expected_counts.append(bin_prob * len(data))
19
20 # Розрахунок статистики хі-квадрат
21 expected_counts = np.array(expected_counts)
22 chi_squared_stat = np.sum((observed_counts - expected_counts) ** 2 / expected_counts)
23
24 # Ступені свободи = (кількість бінів - 1 - кількість оцінених параметрів)
25 degrees_of_freedom = num_bins - 1 - 2
26
27 # Знаходження критичного значення для розподілу хі-квадрат
28 critical_value = chi2.ppf(0.95, degrees_of_freedom)
29
30 # Перевірка, чи входить статистика хі-квадрат в критичний діапазон
31 return chi_squared_stat < critical_value
```

Нижче на рисунку 4.36 перевірка показала розподіл є нормальним:


```
time_in_system_stable_filtered_is_normal = chi_squared_normality_test(
    time_in_system_stable_filtered['timeInSystem'],
    time_in_system_stable_filtered mean,
    time_in_system_stable_filtered std dev
)
time_in_system_stable_filtered_is_normal
```

✓ 0.0s

True

Рисунок 4.36 — Результат перевірки

На рисунку поверх гістограми побудуємо графік нормального розподілу функцією `norm.pdf`, що надає бібліотека `Scipy` [7].

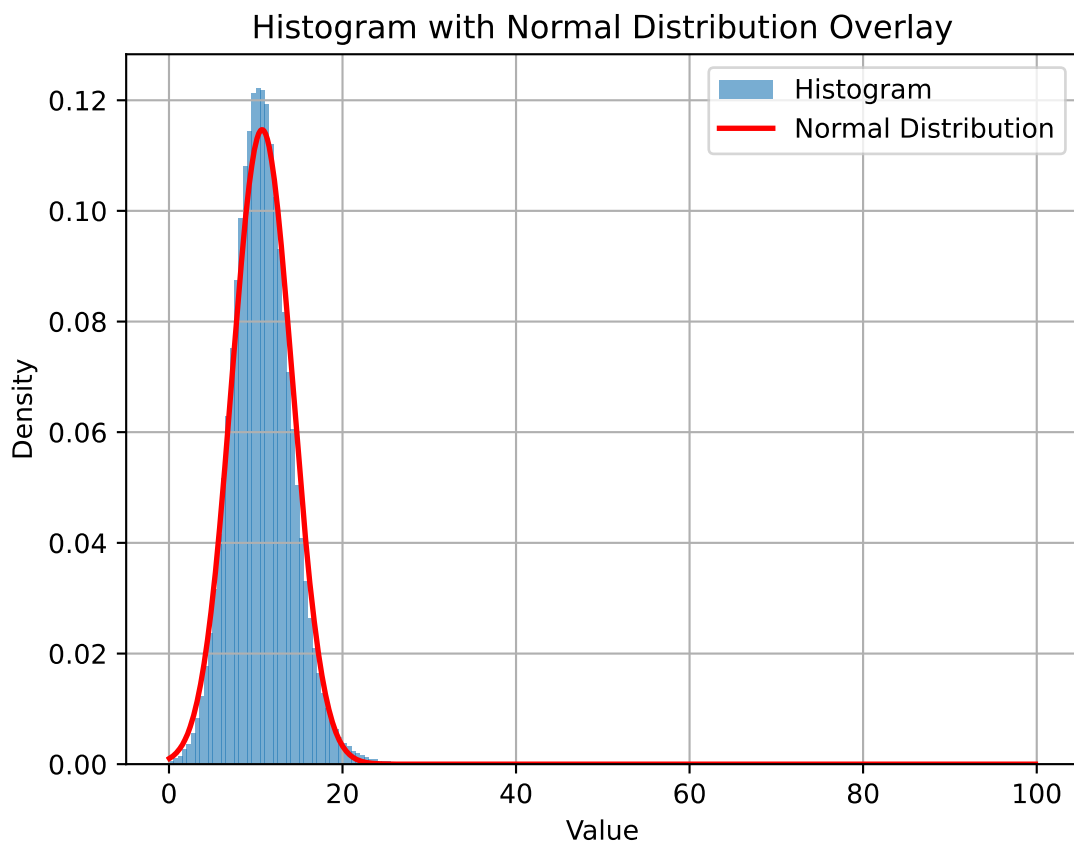


Рисунок 4.37 — Гістограма розподілу часу завдання в системі із нормальним розподілом з ідентичними середнім значенням та середньоквадратичним відхиленням

Нижче можна розглянути код для побудови:

```
1 from scipy.stats import norm
```

```

2
3 count, bins, _ = plt.hist(time_in_system_stable_filtered['timeInSystem'], bins=200,
density=True, alpha=0.6, label='Histogram')
4 x = np.linspace(bins[0], bins[-1], 1000)
5 pdf = norm.pdf(
6     x,
7     time_in_system_stable_filtered_mean,
8     time_in_system_stable_filtered_std_dev
9 )
10 plt.plot(x, pdf, 'r-', lw=2, label='Normal Distribution')
11 plt.title('Histogram with Normal Distribution Overlay')
12 plt.xlabel('Value')
13 plt.ylabel('Density')
14 plt.legend()
15 plt.grid(True)
16 plt.savefig(fname=f'histNormTimeInSystem.svg', format='svg')
17 plt.show()

```

4.4.4 Аналіз розподілу часу очікування виділення пам'яті

Припустимо, що величина розподілена експоненційно. Аналогічно до пункту 4.4.3 напишемо функцію перевірки:

```

1 def chi_squared_exponential_test(data: Sequence[float], mean: float, alpha: float = 0.05) ->
bool:
2     # Sort data and calculate the number of bins
3     sorted_data = np.sort(data)
4     n = len(data)
5     bin_width = 2 / np.sqrt(n) # Rule of thumb for bin width in exponential distribution
6     bins = int(np.ceil(1 / bin_width))
7     # Create bin edges and calculate expected frequencies
8     max_data = max(sorted_data)
9     bin_edges = np.linspace(0, max_data, bins + 1)
10    observed_counts, _ = np.histogram(sorted_data, bins=bin_edges)
11    # Calculate expected frequencies for exponential distribution
12    expected_counts = np.diff(len(data) * (1 - np.exp(-bin_edges / mean)))

```

```

13  # Calculate chi-squared statistic
14  chi_squared_stat = np.sum((observed_counts - expected_counts)**2 / expected_counts)
15  # Degrees of freedom
16  degrees_of_freedom = bins - 1
17  # Compute the p-value
18  p_value = 1 - chi2.cdf(chi_squared_stat, df=degrees_of_freedom)
19  return p_value < alpha

```

На рисунках 4.38, 4.39 й справді бачимо, що величина розподілена експоненційно.

```

chi_squared_exponential_test(
    time_wait_allocate_stable_filtered['timeWaitAllocate'],
    time_wait_allocate_stable_filtered_mean,
    0.05
)
✓ 0.0s
True

```

Рисунок 4.38 — Результат перевірки

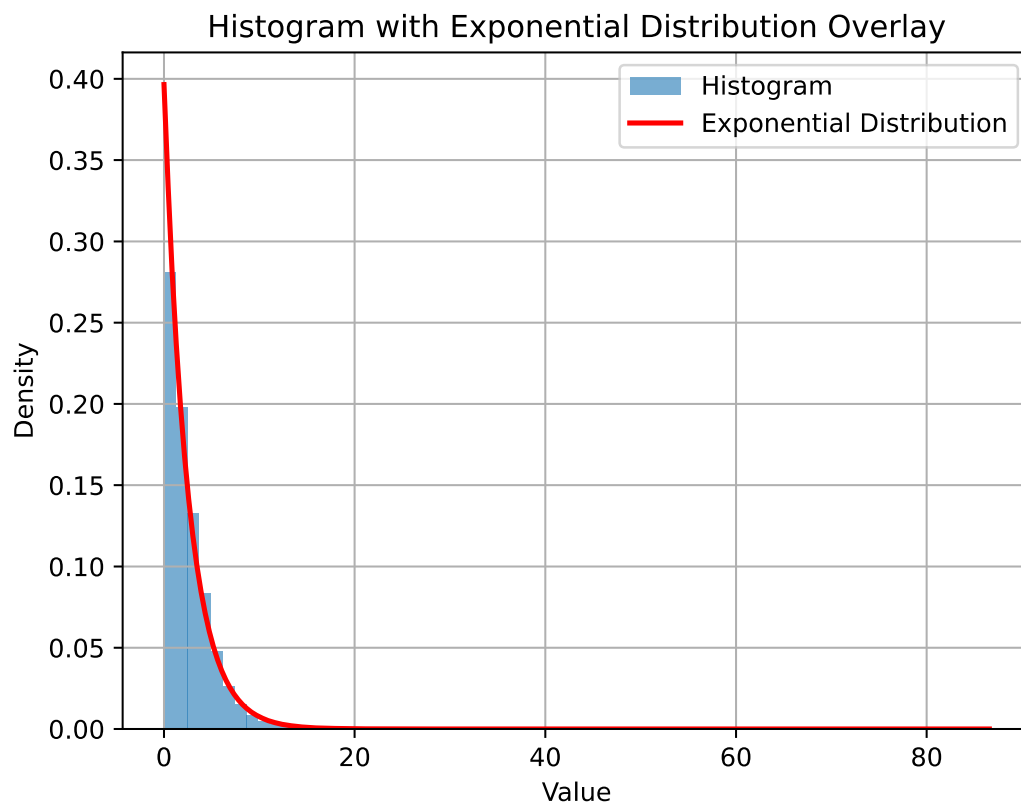


Рисунок 4.39 — Гістограма розподілу часу очікування пам'яті із експоненційним розподілом з ідентичним середнім значенням

4.4.5 Аналіз розподілу кількості зайнятих сторінок

З рисунку 4.33 бачимо, що розподіл не є типовим, змішаним та складається з декількох частин:

- інтервал 0: дельта-розподіл (Delta distribution), оскільки всі значення сконцентровані на нулі;
- інтервал 20–40: рівномірний розподіл (Uniform distribution) або близький до рівномірного, якщо значення розподілені приблизно рівномірно;
- інтервал 40–60: модальний розподіл (Unimodal distribution), де значення концентруються навколо одного піку;
- інтервал 60–80: правосторонньо скошений розподіл (Right-skewed distribution), де значення поступово зменшуються до кінця інтервалу;
- інтервал 80–120: правосторонньо скошений розподіл (Right-skewed distribution), що триває з більш різким спадом частоти;
- інтервал 120–131: дельта-розподіл або скінченний розподіл (Finite distribution), якщо всі значення концентруються в цьому граничному стані.

4.5 Висновки до розділу

У даному розділі були визначені перехідні періоди, середні значення та середньоквадратичні відхилення, типи розподілів вихідних параметрів моделі. У наступному розділі результати будуть інтерпретовані.

5 ІНТЕРПРЕТАЦІЯ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ ТА ЕКСПЕРИМЕНТІВ

У цьому розділі проаналізовано отримані дані моделювання багатопроцесорної обчислювальної системи, зокрема середні значення, середньоквадратичні відхилення та розподіли основних параметрів системи.

5.1 Час виконання завдань у системі

Середній час перебування завдань у системі склав 10.10 с, а середньоквадратичне відхилення — 3.31 с. Це свідчить про стабільність роботи системи у більшості сценаріїв. Гістограма розподілу часу вказує на схожість до нормального розподілу, що підтверджено статистичним тестом згоди χ^2 на рисунку 4.36.

5.2 Завантаження компонентів системи

Завантаження процесорів: середнє значення — 0.714, середньоквадратичне відхилення — 0.00023. Це свідчить про оптимальне використання ресурсів у більшості сценаріїв роботи системи.

Завантаження каналу передачі даних: середнє — 0.0021, відхилення — $5.83e-07$. Такі низькі значення навантаження вказують на резерв пропускну здатності каналу.

Завантаження дисків: середнє — 0.00134, відхилення — $5.16e-07$, що також демонструє низький рівень використання.

5.3 Використання пам'яті

Середнє значення використання сторінок пам'яті склало 55.12, а середньоквадратичне відхилення — 30.54, що свідчить про нерівномірність розподілу ресурсів між завданнями. Середній час очікування виділення пам'яті становив 2.17 с, при цьому відхилення досягло 2.02 с, що вказує на можливі пікові навантаження.

5.4 Кількість завдань в очікуванні пам'яті

Середня кількість завдань, що очікували виділення пам'яті, була низькою — 0.021, зі значним відхиленням 0.15, що свідчить про рідкісні випадки перевантаження пам'яті. Цей показник є прийнятним для ефективної роботи системи.

5.5 Висновки до розділу

Отримані результати демонструють, що система ефективно працює у більшості сценаріїв, забезпечуючи стабільне виконання завдань і оптимальне використання ресурсів. Проте аналіз розподілу часу очікування та використання пам'яті показав можливі перевантаження у пікові періоди.

ВИСНОВКИ

У ході виконання курсової роботи було досягнуто поставленої мети — досліджено характеристики роботи багатопроцесорної обчислювальної системи за допомогою імітаційного моделювання на основі мереж Петрі. Основні результати роботи полягають у наступному:

1. Розроблено концептуальну модель багатопроцесорної обчислювальної системи, яка дозволяє ефективно аналізувати її роботу та взаємодію компонентів.
2. Побудовано формалізовану модель системи з використанням мереж Петрі, яка включає всі необхідні параметри та характеристики для моделювання.
3. Реалізовано програмну модель у середовищі PetriObjModelPaint, що забезпечує імітацію роботи системи та збір статистичних даних для аналізу.
4. Проведено експериментальне дослідження, у результаті якого визначено:
 - середній час виконання завдань у системі та середньоквадратичне відхилення;
 - середнє завантаження процесорів, дисків, каналу передачі даних;
 - середній час очікування виділення пам'яті;
 - кількість завдань, що очікували ресурсу, та розподіли навантаження системи.
5. Результати моделювання показали, що система є стабільною в більшості сценаріїв, проте виявлені моменти нерівномірного розподілу пам'яті та часу очікування ресурсів вимагають додаткової оптимізації.

Робота демонструє, що використання мереж Петрі є ефективним інструментом для аналізу систем та їхніх характеристик.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Business Process Model and Notation [Електронний ресурс] — <https://www.bpmn.org/>
2. Стеценко, І. В. Моделювання систем: навчальний посібник, 2011. 407
3. PetriObjModelPaint [Електронний ресурс] — <https://github.com/StetsenkoInna/PetriObjModelPaint.git>
4. Python [Електронний ресурс] — <https://www.python.org/>
5. Pandas [Електронний ресурс] — <https://numpy.org/>
6. NumPy [Електронний ресурс] — <https://numpy.org/>
7. Scipy [Електронний ресурс] — <https://scipy.org/>

ДОДАТОК А

Результати верифікації

Таблиця А.1 Набори параметрів

Індекс	Кількість сторінок	Кількість процесорів	Кількість дисків	Початок сторінок	Кінець сторінок	Середій інтервал надходження завдань, с
0	200	4	5	30	40	8
1	400	5	11	70	100	8
2	700	12	12	30	70	8
3	1000	30	30	70	80	15
4	1000	40	8	60	100	9
5	131	2	4	20	60	7
6	131	2	4	20	60	5

Таблиця А.2 Середні значення вихідних параметрів

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	0.0009360103 51192126	0.0018732614 053677868	0.3125498430 1379857	0.0	43.287725734 20972	9.4950221388 01987	0.0
0	0.0009367716 927609831	0.0018736703 00544177	0.3127120947 7045736	0.0	43.317718294 397615	9.4912812737 42294	0.0
0	0.0009420197 845173965	0.0018796318 284126626	0.3129606246 348914	0.0	43.273784112 76173	9.4782467442 35886	0.0
0	0.0009339838 342392239	0.0018721119 92227919	0.3123017246 403013	0.0	43.282100480 21178	9.4885524146 42628	0.0
0	0.0009377930 537845769	0.0018743479 865269846	0.3116466958 5077337	0.0	43.220633891 72833	9.4646266331 12106	0.0
1	0.0004253035 7000574435	0.0018702121 521688366	0.2495543875 669563	4.5865727982 77897e-06	105.60035325 188251	9.5064705645 11407	0.8532533290 942025

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
1	0.0004258024 2580648575	0.0018735048 455757574	0.2496723413 5640504	4.9934105970 36274e-06	105.57352457 280489	9.4863905541 17698	0.5419994385 3752
1	0.0004258038 8223467896	0.0018764226 34409976	0.2506825031 814229	7.8180997302 20668e-06	105.66030069 191149	9.5067717984 30059	3.2325037051
1	0.0004256360 282428279	0.0018753591 680659943	0.2506357934 664264	1.8191421722 579236e-06	106.03298739 821577	9.5115185212 03582	0.4548968775
1	0.0004252183 283449599	0.0018708666 37860342	0.2493496544 6656035	4.0670657529 48077e-06	105.41275631 178681	9.4749327114 85453	1.6002036635
2	0.0003905338 267344237	0.0018729459 449796592	0.1043419975 1852612	0.0	61.922011326 29012	9.5122580844 26787	0.0
2	0.0003905448 4046708443	0.0018799245 869985232	0.1040531302 5806305	0.0	61.642261245 657934	9.4902733669 94723	0.0
2	0.0003904902 7868580686	0.0018655163 2232557	0.1040250364 7106194	0.0	61.769947622 47749	9.4928812531 54808	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
2	0.0003906049 316387162	0.0018751445 573874908	0.1044183618 399701	0.0	62.110648431 79735	9.5131938583 39884	0.0
2	0.0003895007 918218604	0.0018748490 981729542	0.1041926404 5247213	0.0	62.005298003 90665	9.4972105785 48748	0.0
3	8.3252234109 88142e-05	0.0009980202 14955883	0.0221948596 35989744	0.0	49.796811295 159415	9.5001365321 89162	0.0
3	8.3350970328 90894e-05	0.0009999187 538115863	0.0223503768 87860658	0.0	49.922044096 680466	9.5274672325 01463	0.0
3	8.3640575120 87945e-05	0.0010010261 740462866	0.0223119888 55716215	0.0	49.790239399 21938	9.5073177792 55189	0.0
3	8.3277942063 46052e-05	0.0009990816 740021642	0.0222545214 44987567	0.0	49.686900834 1537	9.5001338904 04134	0.0
3	8.3134654527 76243e-05	0.0009974918 853889287	0.0222160478 75367916	0.0	49.642125413 772376	9.4822541602 84727	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
4	0.0005194555 476797408	0.0016589458 545388863	0.0277566178 03047664	0.0	88.235206601 72791	9.4719273559 6006	0.0
4	0.0005211279 19098321	0.0016697866 173494157	0.0278495112 1980277	0.0	88.432927524 98242	9.4739563069 1728	0.0
4	0.0005217099 928487044	0.0016737284 459259927	0.0278370416 64965367	0.0	88.588028489 2275	9.4866561790 88905	0.0
4	0.0005208774 785046074	0.0016658098 663149548	0.0278557223 20352855	0.0	88.567865309 99608	9.4917136441 97953	0.0
4	0.0005216014 084532069	0.0016631591 479217577	0.0278197887 18339046	0.0	88.495374156 16511	9.4666685570 0917	0.0
5	0.0013361316 751302958	0.0021385743 451527842	0.7136300664 160912	0.0308802467 67672676	58.920925943 38616	10.152168042 183192	2.2130075021 738915
5	0.0013345221 023116213	0.0021335454 321360363	0.7136058304 54851	0.0325866718 26679514	58.909500503 36583	10.193556093 065643	2.2368560465 532776

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
5	0.0013428582 497630513	0.0021512103 98355921	0.7161423346 712389	0.0317071973 0101972	59.009528742 252144	10.177243529 153868	2.2865871453 8215
5	0.0013365079 583489167	0.0021399357 087336806	0.7151894300 683876	0.0314686938 80982515	59.081004164 534455	10.177539762 40193	2.2880328742 63237
5	0.0013376912 744278067	0.0021349486 14874355	0.7129872073 262552	0.0309275669 50830883	58.944502658 95246	10.161993312 02116	2.2640942311 3396
6	0.0015549157 397447115	0.0024838855 69098083	0.8287059713 231449	13380.920708 126088	113.31131969 782183	3308.7492687 70184	4594.3850397 75223
6	0.0015543881 269259644	0.0024900719 739375694	0.8294516915 77259	13780.984081 833849	113.22666715 306929	2813.8049376 168924	4028.6534393 981706
6	0.0015629435 068970977	0.0025014566 3483877	0.8317256221 913297	13257.518796 66013	113.21090187 914979	3044.6303227 318176	4279.4427462 95762
6	0.0015511359 242988724	0.0024889314 250281804	0.8284132476 642948	13564.411614 174862	112.76750700 449382	3461.1957975 910814	4760.6365120 93716

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
6	0.0015561442 599845425	0.0024943175 27309956	0.8307107776 604068	13360.958533 671577	112.66502143 440532	2532.9424660 90692	3496.3109908 940405

Таблиця А.3 Середньоквадратичні відхилення

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	6.0633182301 23687e-06	1.2187990004 602001e-05	0.0014999733 399472307	0.0	22.991929613 833783	2.8910147036 59474	0.0
0	6.7021406918 88731e-06	1.3241064730 359894e-05	0.0014503993 806075325	0.0	23.069494401 712703	2.8975417294 92824	0.0
0	6.1765182491 57647e-06	1.2569706992 329e-05	0.0014247731 34591655	0.0	23.059354995 769432	2.8928774788 811773	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	6.9354349774 36626e-06	1.1943064236 362212e-05	0.0019805472 416211463	0.0	23.051140418 283836	2.8999232260 332723	0.0
0	6.2839684845 80637e-06	1.5880106291 393236e-05	0.0020546912 403766777	0.0	23.085492335 510185	2.8985053307 85881	0.0
1	2.4377928324 90952e-06	1.0970839091 620434e-05	0.0014581422 307217745	0.0021416236 27444322	56.448500048 16332	2.8941574272 191426	0.0245082831 94041885
1	2.5645985147 086683e-06	1.3916728122 596208e-05	0.0015592548 667633431	0.0022345884 773011127	56.554545567 76581	2.9129551636 906164	0.5639463883 385167
1	3.6584033561 87013e-06	1.4010064319 879976e-05	0.0019143012 744062922	0.0027960755 725726138	56.546243338 906606	2.9006892141 046845	0.0
1	2.4111802958 882988e-06	1.0844180688 49111e-05	0.0010754011 045203766	0.0013487545 599477123	56.628929775 15004	2.8919202689 902477	0.0
1	3.1851392422 796374e-06	1.6759857050 14997e-05	0.0012268877 109490733	0.0020166926 419075406	56.477288115 50977	2.9034802217 458116	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
2	3.1314056490 381993e-06	1.5902191783 644085e-05	0.0004627937 618526907	0.0	35.117783525 69744	2.9096346241 721545	0.0
2	2.7605322722 03941e-06	1.5204369363 794972e-05	0.0005330774 888740822	0.0	34.986283728 69827	2.9094407389 289776	0.0
2	2.9470521527 877146e-06	1.4571256543 55262e-05	0.0006151935 78240626	0.0	35.117482245 891864	2.9071993464 97265	0.0
2	2.9425100298 434966e-06	1.2549401709 395428e-05	0.0004933064 093384896	0.0	35.041043982 0899	2.8988637314 95643	0.0
2	2.8240482434 28334e-06	1.0976999824 895244e-05	0.0004376764 30610689	0.0	35.127839194 62015	2.9035292032 11126	0.0
3	1.0313923455 073889e-06	6.5733714509 621685e-06	0.0001639139 2587692703	0.0	38.875647459 608174	2.9476198122 015806	0.0
3	7.7502613994 92419e-07	9.5037515492 1653e-06	0.0001514501 4808299955	0.0	38.851055286 01111	2.9444798322 524015	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
3	5.9938053088 3507e-07	6.6561218656 18153e-06	0.0001454151 510442333	0.0	38.820826724 05772	2.9499255758 929386	0.0
3	6.1409622276 62226e-07	7.2624227376 45955e-06	0.0001471072 9220316263	0.0	38.826815300 71401	2.9346540949 157616	0.0
3	6.3682041571 59772e-07	7.3360886716 537706e-06	0.0001203109 6040425972	0.0	38.831770776 875835	2.9253090518 378593	0.0
4	3.7020091998 908736e-06	1.5498275789 066978e-05	0.0001692764 9378458193	0.0	50.841411307 17622	2.9177876646 131997	0.0
4	3.0464706405 17444e-06	1.0673961771 097588e-05	0.0001768862 5442550772	0.0	50.787061694 530145	2.9030262732 416343	0.0
4	2.5898961682 991903e-06	1.0468779241 768465e-05	0.0001354195 042958765	0.0	50.867016915 48756	2.9108177033 25894	0.0
4	4.8542137828 12781e-06	1.4388571375 9911e-05	0.0001402846 2885328614	0.0	50.669218319 81429	2.9129497336 68332	0.0

Індекс набору параметрів	Завантаженн я дисків	Завантаженн я каналу введення- виведення	Завантаженн я процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
4	3.4582887819 867115e-06	1.2168680528 818852e-05	0.0001327979 7871041873	0.0	50.807847019 563916	2.9149022870 38537	0.0
5	1.0429539252 64986e-05	1.5452226586 13835e-05	0.0027240130 9799634	0.1839526487 0787148	30.238845100 109078	3.4093331256 802855	2.2837317803 614865
5	9.8511164922 73075e-06	1.5284838346 721153e-05	0.0039473925 11602092	0.1900819280 90188	30.346295996 055964	3.4220210458 144003	2.3713058029 544203
5	1.0440489039 287137e-05	1.4810999924 472431e-05	0.0036651459 241224317	0.1838350889 937267	30.307264891 898427	3.3604252845 9727	2.2539288393 29754
5	1.1500476639 141201e-05	1.6151307410 099243e-05	0.0036823174 25735091	0.1838137642 750485	30.350028590 139242	3.3982026023 483685	2.4833034772 36511
5	6.7932165172 25891e-06	1.3808350765 006422e-05	0.0033327505 556365695	0.1823208954 55085	30.310882909 16066	3.3826301229 363542	2.3267677675 503418
6	1.6487713608 485905e-05	2.7717817585 898182e-05	0.0066449176 177217914	7676.5059956 05145	12.399661375 35893	33001.993335 012055	38936.049733 319946

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
6	1.2088566310 81339e-05	1.7969743858 613083e-05	0.0045895362 39650023	7870.6757237 16892	12.404640705 357226	32836.597045 124334	39690.970024 59861
6	1.4024173231 770203e-05	2.1239429954 140986e-05	0.0057928199 01398434	7697.7152225 12689	12.397876864 875949	30225.365413 47308	35935.396834 251136
6	1.1495327235 395708e-05	1.5410940009 045226e-05	0.0044120521 08418184	7764.8413438 40343	12.510213152 973288	37113.607723 822264	43509.778989 32139
6	1.0685274560 545559e-05	1.4690329046 40187e-05	0.0062500224 88174315	7738.0275167 55956	12.502708890 985504	27056.371631 144702	32030.925718 917628

Таблиця А.4 Глобальні середні значення

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	0.0009373157 432988612	0.0018746047 02615906	0.3124341965 820444	0.0	43.276392502 66184	9.4835458409 0698	0.0
1	0.0004255528 469269394	0.0018732730 876161812	0.2499789360 0755422	4.6568582101 48168e-06	105.65598444 532029	9.4972168299 4964	1.3365714027 463445
2	0.0003903349 3386957834	0.0018736761 019728394	0.1042062333 0801868	0.0	61.890033326 025915	9.5011634282 92989	0.0
3	8.3331275230 17854e-05	0.0009991077 404409698	0.0222655589 3998442	0.0	49.767624207 79706	9.5034619189 26934	0.0
4	0.0005209544 693169161	0.0016662859 864102013	0.0278237363 45301543	0.0	88.463880416 4198	9.4781844086 34673	0.0
5	0.0013375422 519963382	0.0021396428 99850555	0.7143109737 873649	0.0315140753 4543706	58.973092402 49821	10.172500147 76516	2.2577155599 013032

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
6	0.0015559055 115702378	0.0024917326 260425116	0.8298014620 83287	13468.958746 893302	113.03628343 3788	3032.2645585 601335	4231.8857456 91383

Таблиця А.5 Глобальні середньоквадратичні відхилення

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	6.4322761266 37465e-06	1.3164386451 00927e-05	0.0016820768 674288482	0.0	23.051482353 02199	2.8959724937 705253	0.0
1	2.8514228483 10914e-06	1.3300333854 54754e-05	0.0014467974 37472172	0.0021075469 7583466	56.531101369 09911	2.9006404591 501003	0.1176909343 0651172

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
2	2.9211096694 60337e-06	1.3840843845 05647e-05	0.0005084095 337833155	0.0	35.078086535 39953	2.9057335288 610333	0.0
3	7.3134313096 44675e-07	7.4663512550 19315e-06	0.0001456394 9552231644	0.0	38.841223109 45337	2.9403976734 201085	0.0
4	3.5301757147 013995e-06	1.2639653741 348597e-05	0.0001509329 720139342	0.0	50.794511051 31443	2.9118967323 775196	0.0
5	9.8029675881 15433e-06	1.5101544606 48752e-05	0.0034703239 030185046	0.1848008651 0438393	30.310663497 472678	3.3945224362 753357	2.3438075334 865025
6	1.2956210989 40215e-05	1.9405652090 81987e-05	0.0055378696 71072549	7749.5531604 86204	12.443020197 910178	32046.787029 715284	38020.624260 08174

Таблиця А.6 Відсоткові відхилення середніх значень відносно глобальних середніх

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	0.1392691967 533767	0.0716576271 3838489	0.0370146523 7137983	0.0	0.0261880228 28342115	0.1210127318 1496974	0.0
0	0.0580434652 6426225	0.0498452858 04791147	0.0889461497 6628373	0.0	0.0954926909 2434332	0.0815668840 0182767	0.0
0	0.5018630330 457753	0.2681699128 2170133	0.1684924565 2555628	0.0	0.0060272812 71077231	0.0558767444 1597132	0.0
0	0.3554734979 6032957	0.1329725880 0794026	0.0423999495 5491644	0.0	0.0131895872 54963123	0.0527922131 61996636	0.0
0	0.0509231269 3220478	0.0136944118 70573433	0.2520533091 083212	0.0	0.1288430197 36637	0.1994950845 628602	0.0
1	0.0585771950 5230517	0.1634003855 3801017	0.1698336857 410604	1.5092882088 852497	0.0526531400 27829275	0.0974362776 7437438	36.161036564 07996

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
1	0.0586481517 7448804	0.0123718192 02888379	0.1226481943 0222648	7.2270267141 63101	0.0780456241 5305075	0.1139941945 7078315	59.448523481 511224
1	0.0589903955 6482662	0.1681306807 1151702	0.2814505834 393168	67.883568221 67273	0.0040851889 40183183	0.1006080902 595303	141.85043151 888138
1	0.0195466477 28761482	0.1113601889 4435572	0.2627651230 7913946	60.936277417 81635	0.3568212012 5489875	0.1505882355 8541926	65.965389012 1177
1	0.0786080000 157837	0.1284623033 2073935	0.2517338264 752249	12.665029309 13427	0.2302076260 14175	0.2346384089 4854077	19.724517538 827513
2	0.0509544105 81093804	0.0389692216 4996266	0.1302841549 8537582	0.0	0.0516690629 26091594	0.1167715534 7901642	0.0
2	0.0537760213 83785615	0.3334880035 618026	0.1469231207 1493263	0.0	0.4003424575 048457	0.1146181873 4574075	0.0
2	0.0397978255 9775387	0.4354957422 298137	0.1738829158 33972	0.0	0.1940307624 5869122	0.0871701155 4098435	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
2	0.0691707930 0621716	0.0783729596 1160305	0.2035660681 87493	0.0	0.3564630586 1442107	0.1266205990 2127817	0.0
2	0.2136990505 6887822	0.0626040007 0640545	0.0130441866 24004125	0.0	0.1862410984 2297834	0.0416038496 135134	0.0
3	0.0948516869 3121654	0.1088496706 6782505	0.3175276407 1741096	0.0	0.0586467363 6114539	0.0349913196 4900005	0.0
3	0.0236347021 883494	0.0811737651 2953402	0.3809378785 6329514	0.0	0.3102818174 29433	0.2525954623 621973	0.0
3	0.3711690356 910503	0.1920146874 7205694	0.2085279595 133558	0.0	0.0454415732 764172	0.0405732180 6672695	0.0
3	0.0640013807 1894364	0.0026089717 60553057	0.0495720544 29917634	0.0	0.1622005770 384334	0.0350191177 8245622	0.0
3	0.2359506702 2915817	0.1617298101 7321286	0.2223661429 293535	0.0	0.2521695500 285194	0.2231582429 9743064	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
4	0.2877260347 0331753	0.4405085280 185574	0.2412276389 5156238	0.0	0.2584939905 591612	0.0660153084 6891568	0.0
4	0.0332946143 3210878	0.2100858416 721115	0.0926362806 9700314	0.0	0.0349892987 8689205	0.0446087724 7272433	0.0
4	0.1450267876 1523216	0.4466495893 5561877	0.0478200321 4342434	0.0	0.1403375843 6019975	0.0893817854 6636275	0.0
4	0.0147787986 94187899	0.0285737321 88213748	0.1149593090 3871425	0.0	0.1175450286 4536483	0.1427407927 5091997	0.0
4	0.1241834314 5018534	0.1876531708 2093308	0.0141879829 27616756	0.0	0.0356006763 40520825	0.1214984972 7562398	0.0
5	0.1054603593 9702729	0.0499407960 9477475	0.0953236610 1887417	2.0112555130 263727	0.0884580695 8198674	0.1998732394 8512992	1.9802342917 531266
5	0.2257984508 6831762	0.2849759515 919465	0.0987165756 0782758	3.4035473656 940165	0.1078320578 7880562	0.2069888915 6673628	0.9239212289 84996

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
5	0.3974452215 45618	0.5406275274 333825	0.2563814572 473852	0.6128117467 061311	0.0617846856 4146393	0.0466294550 9763055	1.2787964079 101573
5	0.0773279233 5178747	0.0136849416 85648728	0.1229795303 8087943	0.1440037949 9352542	0.1829847437 874463	0.0495415538 3204507	1.3428314399 028767
5	0.0111415120 71562979	0.2193957214 3220867	0.1853207510 016095	1.8610998043 802713	0.0484793019 6812992	0.1032866610 1133437	0.2825276729 250885
6	0.0636138774 6016872	0.3149237146 239043	0.1320184176 8170844	0.6536365610 854696	0.2433167967 6546905	9.1180932557 3289	8.5659045604 64851
6	0.0975242155 1242055	0.0666464807 4941076	0.0421511074 64898384	2.3166255150 385484	0.1684270868 590657	7.2045039845 39409	4.8024053225 00249
6	0.4523407928 388308	0.3902508918 7448564	0.2318819857 477694	1.5698314487 891833	0.1544799953 2298093	0.4078062429 208335	1.1237779907 644794
6	0.3065473600 9977024	0.1124198072 0781593	0.1672947665 7066172	0.7086877989 256319	0.2377788981 8149944	14.145574396 536997	12.494448058 780206

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
6	0.0153446602 33500852	0.1037391107 0666274	0.1095823059 6953929	0.8018453040 89568	0.3284449807 6597856	16.466969910 651326	17.381725287 509344

Таблиця А.7 Відсоткові відхилення середніх квадратичних відхилень відносно глобальних відхилень

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	5.7360394555 489185	7.4169536882 017875	10.826112112 223104	0.0	0.2583466792 9890676	0.1711960359 3321298	0.0
0	4.1954754419 46546	0.5824675508 87992	13.773299621 880428	0.0	0.0781383531 6474967	0.0541868310 4463374	0.0
0	3.9761644625 402357	4.5173351670 68703	15.296788025 537554	0.0	0.0341524359 5564912	0.1068730761 7754235	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
0	7.8224075100 79821	9.2774716025 85955	17.744157830 82062	0.0	0.0014833524 929798452	0.1364216086 7360972	0.0
0	2.3056790339 372117	20.629292906 968416	22.152041928 82054	0.0	0.1475392426 7147242	0.0874606723 9257319	0.0
1	14.506091794 311812	17.514558569 750672	0.7841314171 404627	1.6168869306 538798	0.1461165958 8317377	0.2235034649 1606296	79.175725523 41794
1	10.058989804 761875	4.6344270361 14707	7.7728523965 07928	6.0279321373 67418	0.0414713283 4654116	0.4245512228 745629	379.17572552 341795
1	28.300976417 93559	5.3361853401 12885	32.313012507 88344	32.669667847 62997	0.0267852021 99813268	0.0016808341 216654817	100.0
1	15.439399059 434477	18.466853485 91188	25.670237127 368353	36.003582581 42457	0.1730523617 648891	0.3006298189 196342	100.0
1	11.703504240 572592	26.010799679 434946	15.199759194 163525	4.3109043342 2667	0.0951922964 2809489	0.0979012268 3951474	100.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
2	7.1991812487 03803	14.893224442 553528	8.9722495153 0242	0.0	0.1131674906 4362885	0.1342550950 5168806	0.0
2	5.4971368906 5506	9.8514623385 87924	4.8519851520 48742	0.0	0.2617098472 8205143	0.1275825890 819865	0.0
2	0.8881037093 061366	5.2772266393 06617	21.003548785 30306	0.0	0.1123086074 053076	0.0504457006 00984515	0.0
2	0.7326106447 456033	9.3306603998 88158	2.9706611385 58204	0.0	0.1056002677 6900516	0.2364221390 9693718	0.0
2	3.3227587121 004825	20.691253020 5599	13.912623283 491186	0.0	0.1418340170 0207965	0.0758612456 3773718	0.0
3	41.027146060 32164	11.960056171 437603	12.547716049 874943	0.0	0.0886283886 0094871	0.2456177559 50397	0.0
3	5.9729841076 33167	27.287763789 94434	3.9897505411 179797	0.0	0.0253137665 8772318	0.1388301612 8035744	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
3	18.043869490 772856	10.851744871 45215	0.1540409607 1505327	0.0	0.0525122119 30538566	0.3240344855 0371605	0.0
3	16.031723446 096233	2.7313008778 721404	1.0078287318 849428	0.0	0.0370941169 86882454	0.1953333916 7917074	0.0
3	12.924537231 085676	1.7446618691 824116	17.391254362 162773	0.0	0.0243358262 7121429	0.5131490110 5533	0.0
4	4.8675618177 83584	22.616300305 49696	12.153422493 366286	0.0	0.0923333149 4108234	0.2023056714 2641423	0.0
4	13.702011267 302304	15.551786547 9223	17.195237107 752366	0.0	0.0146656727 8649348	0.3046282183 4490844	0.0
4	26.635488496 689263	17.175110521 251586	10.278382192 477798	0.0	0.1427435025 408215	0.0370558831 852656	0.0
4	37.506293598 85491	13.836752734 144923	7.0550145661 11514	0.0	0.2466658875 2781465	0.0361620410 196614	0.0

Індекс набору параметрів	Завантаження дисків	Завантаження каналу введення-виведення	Завантаження процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
4	2.0363556526 46842	3.7261559704 68022	12.015262842 52936	0.0	0.0262547428 3237632	0.1032163890 8408316	0.0
5	6.3916529245 08759	2.3221596782 90004	21.505508588 780994	0.4589894078 868817	0.2369410269 4119	0.4363114306 3531846	2.5631692136 27838
5	0.4911666158 726785	1.2137416735 165762	13.747091681 229847	2.8577046881 361166	0.1175576330 9588185	0.8100877238 342141	1.1732306972 754312
5	6.5033516171 6335	1.9239401636 45726	5.6139434401 0569	0.5226036740 205305	0.0112125739 99029606	1.0044756609 556853	3.8347301505 192575
5	17.316277298 35333	6.9513604797 800115	6.1087532069 32451	0.5341429699 357008	0.1298720916 150403	0.1084148401 467299	5.9516808337 28826
5	30.702448455 898143	8.5633216679 40844	3.9642797394 869422	1.3419686362 929737	0.0007238762 292388592	0.3503383336 6057726	0.7270121668 571244
6	27.257217576 747035	42.833734502 591156	19.990501987 289175	0.9425984101 059695	0.3484589903 545297	2.9806616944 502404	2.4077076351 40852

Індекс набору параметрів	Завантаженн я дисків	Завантаженн я каналу введення- виведення	Завантаженн я процесорів	Кількість завдань в очікуванні пам'яті	Кількість зайнятих сторінок	Час завдання в системі	Час виділення пам'яті
6	6.6967470605 29283	7.3994330388 210185	17.124516966 807874	1.5629618988 649996	0.3084419372 669509	2.4645528884 89885	4.3932623333 346505
6	8.2428592992 31993	9.4497100882 71713	4.6037600281 13969	0.6689151864 629945	0.3628004480 922638	5.6836325418 62303	5.4844639361 16663
6	11.275547729 204225	20.585301968 10239	20.329434051 782624	0.1972782564 0439292	0.5400051916 2056	15.810697931 773275	14.437308266 40522
6	17.527782086 24543	24.298709583 939484	12.859689003 187388	0.1487265587 0039336	0.4796961840 932415	15.572279972 851051	15.753814298 764059

ДОДАТОК Б

Лістинг програмного коду

```
1 // ./src/main/java/ua/stetsenkoinna/course_work/CourseWorkPetriSim.java
2
3 package ua.stetsenkoinna.course_work;
4
5 import ua.stetsenkoinna.PetriObj.*;
6 import java.util.*;
7 import java.util.function.Consumer;
8
9 public class CourseWorkPetriSim {
10     private final StateTime timeState = new StateTime();
11     private double timeMin;
12     private final PetriP[] listP;
13     private final PetriT[] listT;
14     private PetriT eventMin;
15
16     public CourseWorkPetriSim(PetriNet net) {
17         timeMin = Double.MAX_VALUE;
18         listP = net.getListP();
19         listT = net.getListT();
20         eventMin = this.getEventMin();
21     }
22
23     public void go(
24         final double timeModelling,
25         final Consumer<Double> trackStats
26     ) {
27         setSimulationTime(timeModelling);
28         setTimeCurr(0);
29         input();
30         while (getCurrentTime() < getSimulationTime()) {
31             trackStats.accept(getCurrentTime());
```

```

32     setTimeCurr(getTimeMin());
33     output();
34     input();
35 }
36 }
37
38 private void eventMin() {
39     PetriT event = null;
40     double min = Double.MAX_VALUE;
41     for (PetriT transition : listT) {
42         if (transition.getMinTime() < min) {
43             event = transition;
44             min = transition.getMinTime();
45         }
46     }
47     timeMin = min;
48     eventMin = event;
49 }
50
51 private double getTimeMin() {
52     return timeMin;
53 }
54
55 private ArrayList<PetriT> findActiveT() {
56     ArrayList<PetriT> aT = new ArrayList<>();
57
58     for (PetriT transition : listT) {
59         if ((transition.condition(listP)) && (transition.getProbability() != 0)) {
60             aT.add(transition);
61
62         }
63     }
64
65     if (aT.size() > 1) {

```

```

66     aT.sort((o1, o2) -> Integer.compare(o2.getPriority(), o1.getPriority()));
67 }
68     return aT;
69 }
70
71     private double getCurrentTime() {
72         return timeState.getCurrentTime();
73     }
74
75     private void setTimeCurr(double aTimeCurr) {
76         timeState.setCurrentTime(aTimeCurr);
77     }
78
79     private double getSimulationTime() {
80         return timeState.getSimulationTime();
81     }
82
83     private void setSimulationTime(double aTimeMod) {
84         timeState.setSimulationTime(aTimeMod);
85     }
86
87     private void input() {
88         ArrayList<PetriT> activeT = this.findActiveT();
89         if (activeT.isEmpty() && isBufferEmpty()) {
90             timeMin = Double.MAX_VALUE;
91         } else {
92             while (!activeT.isEmpty()) {
93                 doConflict(activeT).actIn(listP, this.getCurrentTime());
94
95                 activeT = this.findActiveT();
96             }
97             this.eventMin();
98         }
99     }

```

```

100
101  private void output() {
102      if (this.getCurrentTime() <= this.getSimulationTime()) {
103          eventMin.actOut(listP, this.getCurrentTime());
104          if (eventMin.getBuffer() > 0) {
105              boolean u = true;
106              while (u) {
107                  eventMin.minEvent();
108                  if (eventMin.getMinTime() == this.getCurrentTime()) {
109                      eventMin.actOut(listP, this.getCurrentTime());
110                  } else {
111                      u = false;
112                  }
113              }
114          }
115          for (PetriT transition : listT) {
116              if (transition.getBuffer() > 0 && transition.getMinTime() ==
this.getCurrentTime()) {
117                  transition.actOut(listP, this.getCurrentTime());
118                  if (transition.getBuffer() > 0) {
119                      boolean u = true;
120                      while (u) {
121                          transition.minEvent();
122                          if (transition.getMinTime() == this.getCurrentTime()) {
123                              transition.actOut(listP, this.getCurrentTime());
124                          } else {
125                              u = false;
126                          }
127                      }
128                  }
129              }
130          }
131      }
132  }

```

```
133
134  private boolean isBufferEmpty() {
135      boolean c = true;
136      for (PetriT e : listT) {
137          if (e.getBuffer() > 0) {
138              c = false;
139              break;
140          }
141      }
142      return c;
143  }
144
145  private PetriT getEventMin() {
146      this.eventMin();
147      return eventMin;
148  }
149
150  private static PetriT doConflict(ArrayList<PetriT> transitions) {
151      PetriT aT = transitions.get(0);
152      if (transitions.size() > 1) {
153          aT = transitions.get(0);
154          int i = 0;
155          while (i < transitions.size() && transitions.get(i).getPriority() == aT.getPriority()) {
156              i++;
157          }
158          if (i != 1) {
159              double r = Math.random();
160              int j = 0;
161              double sum = 0;
162              double prob;
163              while (j < transitions.size() && transitions.get(j).getPriority() == aT.getPriority()) {
164
165                  if (transitions.get(j).getProbability() == 1.0) {
166                      prob = 1.0 / i;
```

```
167         } else {
168             prob = transitions.get(j).getProbability();
169         }
170
171         sum = sum + prob;
172         if (r < sum) {
173             aT = transitions.get(j);
174             break;
175         }
176         else {
177             j++;
178         }
179     }
180 }
181 }
182 return aT;
183 }
184 }
185
186 // ./src/main/java/ua/stetsenkoinna/course_work/CourseWorkNet.java
187
188 package ua.stetsenkoinna.course_work;
189
190 import ua.stetsenkoinna.PetriObj.*;
191
192 import java.util.ArrayList;
193 import java.util.stream.IntStream;
194
195 public class CourseWorkNet {
196     public final PetriP generated_task;
197     public final PetriP generated_io_request;
198     public final PetriP generated_interrupt;
199     public final PetriP processors;
200     public final PetriP pages;
```

```

201  public final PetriP free_disks;
202  public final PetriP total_wait_allocate_task;
203  public final PetriP finished_tasks;
204  public final PetriP is_disk_placement_available;
205
206  public static class TaskObject {
207      public final PetriT generate;
208      public final PetriP task;
209      public final PetriT try_allocate;
210      public final PetriP allocated;
211      public final PetriT fail_allocate;
212      public final PetriP fail_allocate_token;
213      public final PetriT wait_allocate;
214      public final PetriT process;
215      public final PetriT create_io;
216      public final PetriT take_up_disks;
217      public final PetriP busy_disk;
218      public final PetriT place_disk;
219      public final PetriP disk_placed;
220      public final PetriT io_channel_transfer;
221      public final PetriP finish;
222
223
224      public TaskObject(PetriT generate, PetriP task, PetriT tryAllocate, PetriP allocatedTask,
PetriT failAllocateTask, PetriP failAllocateTokenTask, PetriT waitAllocateTask, PetriT
processTask, PetriT createIoTask, PetriT takeUpDisks, PetriP busyDisk, PetriT placeDiskTask,
PetriP diskPlaced, PetriT ioChannelTransfer, PetriP finishedTasks) {
225          this.generate = generate;
226          this.task = task;
227          this.try_allocate = tryAllocate;
228          this.allocated = allocatedTask;
229          this.fail_allocate = failAllocateTask;
230          this.fail_allocate_token = failAllocateTokenTask;
231          this.wait_allocate = waitAllocateTask;

```

```

232     this.process = processTask;
233     this.create_io = createIoTask;
234     this.take_up_disks = takeUpDisks;
235     this.busy_disk = busyDisk;
236     this.place_disk = placeDiskTask;
237     this.disk_placed = diskPlaced;
238     this.io_channel_transfer = ioChannelTransfer;
239     this.finish = finishedTasks;
240 }
241 }
242
243 public final ArrayList<TaskObject> taskObjects;
244 public final PetriNet net;
245
246 public CourseWorkNet(
247     final int pages_num,
248     final int processors_num,
249     final int disk_num,
250     final int pages_start,
251     final int pages_end,
252     final double tasksTimeMean
253 ) throws ExceptionInvalidTimeDelay {
254     final ArrayList<PetriP> d_P = new ArrayList<>();
255     final ArrayList<PetriT> d_T = new ArrayList<>();
256     final ArrayList<ArcIn> d_In = new ArrayList<>();
257     final ArrayList<ArcOut> d_Out = new ArrayList<>();
258
259     generated_task = create_task_generator(d_P, d_T, d_In, d_Out, tasksTimeMean);
260     generated_io_request = create_io_request_generator(d_P, d_T, d_In, d_Out);
261     generated_interrupt = create_interrupt_generator(d_P, d_T, d_In, d_Out);
262     processors = create_processors(processors_num, d_P);
263     pages = create_pages(pages_num, d_P);
264     free_disks = create_free_disks(disk_num, d_P);
265

```

```

266     total_wait_allocate_task = new PetriP("total_wait_allocate_task");
267     d_P.add(total_wait_allocate_task);
268
269     finished_tasks = new PetriP("finished_tasks");
270     d_P.add(finished_tasks);
271
272     is_disk_placement_available = new PetriP("is_disk_placement_available", 1);
273     d_P.add(is_disk_placement_available);
274
275     final double probability = 1.0 / (double) ((pages_end + 1) - pages_start);
276
277     taskObjects = new ArrayList<>();
278
279     IntStream.rangeClosed(pages_start, pages_end).forEach((pages_count) -> {
280         final String task_n_name =
"task_".concat(Integer.toString(pages_count)).concat("_pages");
281         final int priority = (pages_end - pages_count) * 2;
282
283         final PetriT generate_task_n = new PetriT("generate_".concat(task_n_name));
284         generate_task_n.setProbability(probability);
285         d_T.add(generate_task_n);
286         d_In.add(new ArcIn(generated_task, generate_task_n));
287
288         final PetriP task_n_pages = new PetriP(task_n_name);
289         d_P.add(task_n_pages);
290         d_Out.add(new ArcOut(generate_task_n, task_n_pages, 1));
291
292         final PetriT try_allocate_task_n = new PetriT("try_allocate_".concat(task_n_name));
293         d_T.add(try_allocate_task_n);
294         try_allocate_task_n.setPriority(priority - 1);
295         d_In.add(new ArcIn(task_n_pages, try_allocate_task_n));
296         d_In.add(new ArcIn(pages, try_allocate_task_n, pages_count));
297
298         final PetriP allocated_task_n = new PetriP("allocated ".concat(task_n_name));

```

```

299     d_P.add(allocated_task_n);
300     d_Out.add(new ArcOut(try_allocate_task_n, allocated_task_n, 1));
301
302     final PetriT fail_allocate_task_n = new PetriT("fail_allocate_".concat(task_n_name));
303     d_T.add(fail_allocate_task_n);
304     fail_allocate_task_n.setPriority(Integer.MIN_VALUE);
305     d_In.add(new ArcIn(task_n_pages, fail_allocate_task_n));
306     d_Out.add(new ArcOut(fail_allocate_task_n, total_wait_allocate_task, 1));
307
308     final PetriP fail_allocate_token_task_n = new
PetriP("fail_allocate_token_".concat(task_n_name));
309     d_P.add(fail_allocate_token_task_n);
310     d_Out.add(new ArcOut(fail_allocate_task_n, fail_allocate_token_task_n, 1));
311
312     final PetriT wait_allocate_task_n = new
PetriT("wait_allocate_".concat(task_n_name));
313     d_T.add(wait_allocate_task_n);
314     wait_allocate_task_n.setPriority(priority);
315     d_In.add(new ArcIn(fail_allocate_token_task_n, wait_allocate_task_n));
316     d_In.add(new ArcIn(total_wait_allocate_task, wait_allocate_task_n));
317     d_In.add(new ArcIn(pages, wait_allocate_task_n, pages_count));
318     d_Out.add(new ArcOut(wait_allocate_task_n, allocated_task_n, 1));
319
320     final PetriT process_task_n = new PetriT("process_".concat(task_n_name), 10.0);
321     process_task_n.setDistribution("norm", process_task_n.getTimeServ());
322     process_task_n.setParamDeviation(3.0);
323     process_task_n.setPriority(priority);
324     d_T.add(process_task_n);
325     d_In.add(new ArcIn(allocated_task_n, process_task_n));
326     d_In.add(new ArcIn(processors, process_task_n));
327
328     final PetriP processed_task_n = new PetriP("processed_".concat(task_n_name));
329     d_P.add(processed_task_n);
330     d_Out.add(new ArcOut(process_task_n, processed_task_n, 1));

```

```

331
332     final PetriT create_io_task_n = new PetriT("create_io_".concat(task_n_name));
333     create_io_task_n.setPriority(priority);
334     d_T.add(create_io_task_n);
335     d_In.add(new ArcIn(processed_task_n, create_io_task_n));
336     d_In.add(new ArcIn(generated_io_request, create_io_task_n));
337
338     final PetriP io_task_n = new PetriP("io_".concat(task_n_name));
339     d_P.add(io_task_n);
340     d_Out.add(new ArcOut(create_io_task_n, io_task_n, 1));
341
342     final PetriT take_up_disks_task_n = new
PetriT("take_up_disks_".concat(task_n_name));
343     take_up_disks_task_n.setPriority(priority);
344     d_T.add(take_up_disks_task_n);
345     d_In.add(new ArcIn(io_task_n, take_up_disks_task_n));
346     d_In.add(new ArcIn(generated_interrupt, take_up_disks_task_n));
347     d_In.add(new ArcIn(free_disks, take_up_disks_task_n));
348     d_Out.add(new ArcOut(take_up_disks_task_n, processors, 1));
349     d_Out.add(new ArcOut(take_up_disks_task_n, pages, pages_count));
350
351     final PetriP busy_disk_task_n = new PetriP("busy_disk_".concat(task_n_name));
352     d_P.add(busy_disk_task_n);
353     d_Out.add(new ArcOut(take_up_disks_task_n, busy_disk_task_n, 1));
354
355     final PetriT place_disk_task_n = new PetriT("place_disk_".concat(task_n_name),
0.0375);
356     place_disk_task_n.setDistribution("unif", place_disk_task_n.getTimeServ());
357     place_disk_task_n.setParamDeviation(0.021650635);
358     d_T.add(place_disk_task_n);
359     d_In.add(new ArcIn(is_disk_placement_available, place_disk_task_n));
360     d_In.add(new ArcIn(busy_disk_task_n, place_disk_task_n));
361
362     final PetriP disk_placed_task_n = new PetriP("disk_placed_".concat(task_n_name));

```

```

363     d_P.add(disk_placed_task_n);
364     d_Out.add(new ArcOut(place_disk_task_n, disk_placed_task_n, 1));
365
366     final PetriT io_channel_transfer_task_n = new
PetriT("io_channel_transfer_".concat(task_n_name), 0.015);
367     io_channel_transfer_task_n.setDistribution("unif",
io_channel_transfer_task_n.getTimeServ());
368     io_channel_transfer_task_n.setParamDeviation(0.007216878);
369     d_T.add(io_channel_transfer_task_n);
370     d_In.add(new ArcIn(disk_placed_task_n, io_channel_transfer_task_n));
371     d_Out.add(new ArcOut(io_channel_transfer_task_n, is_disk_placement_available,
1));
372     d_Out.add(new ArcOut(io_channel_transfer_task_n, free_disks, 1));
373
374     final PetriP finish_task_n = new PetriP("finished_tasks_".concat(task_n_name));
375     d_P.add(finish_task_n);
376     d_Out.add(new ArcOut(io_channel_transfer_task_n, finish_task_n, 1));
377     d_Out.add(new ArcOut(io_channel_transfer_task_n, finished_tasks, 1));
378
379     taskObjects.add(new TaskObject(
380         generate_task_n,
381         task_n_pages,
382         try_allocate_task_n,
383         allocated_task_n,
384         fail_allocate_task_n,
385         fail_allocate_token_task_n,
386         wait_allocate_task_n,
387         process_task_n,
388         create_io_task_n,
389         take_up_disks_task_n,
390         busy_disk_task_n,
391         place_disk_task_n,
392         disk_placed_task_n,
393         io_channel_transfer_task_n,

```

```

394         finish_task_n
395     ));
396 });
397 for(final PetriT transition : d_T) {
398     transition.setMoments(true);
399 }
400 net = new PetriNet("CourseWork", d_P, d_T, d_In, d_Out);
401 PetriP.initNext();
402 PetriT.initNext();
403 }
404
405 private static PetriP create_task_generator(
406     ArrayList<PetriP> d_P,
407     ArrayList<PetriT> d_T,
408     ArrayList<ArcIn> d_In,
409     ArrayList<ArcOut> d_Out,
410     final double tasksTimeMean
411 ) {
412     final PetriP task_generator = new PetriP("generator_task", 1);
413     d_P.add(task_generator);
414     final PetriT generate_task = new PetriT("generate_task", tasksTimeMean);
415     generate_task.setDistribution("poisson", generate_task.getTimeServ());
416     d_T.add(generate_task);
417     d_In.add(new ArcIn(task_generator, generate_task, 1));
418     d_Out.add(new ArcOut(generate_task, task_generator, 1));
419     final PetriP generated_task = new PetriP("generated_task", 0);
420     d_P.add(generated_task);
421     d_Out.add(new ArcOut(generate_task, generated_task, 1));
422     return generated_task;
423 }
424
425 private static PetriP create_processors(
426     final int num,
427     ArrayList<PetriP> d_P

```

```

428     ) {
429         final PetriP processors = new PetriP("processors", num);
430         d_P.add(processors);
431         return processors;
432     }
433
434     private static PetriP create_pages(
435         final int num,
436         ArrayList<PetriP> d_P
437     ) {
438         final PetriP pages = new PetriP("pages", num);
439         d_P.add(pages);
440         return pages;
441     }
442
443     private static PetriP create_free_disks(
444         final int num,
445         ArrayList<PetriP> d_P
446     ) {
447         final PetriP disks = new PetriP("free_disks", num);
448         d_P.add(disks);
449         return disks;
450     }
451
452     private static PetriP create_io_request_generator(ArrayList<PetriP> d_P,
ArrayList<PetriT> d_T, ArrayList<ArcIn> d_In, ArrayList<ArcOut> d_Out) {
453         final PetriP generator_io_request = new PetriP("generator_io_request", 1);
454         d_P.add(generator_io_request);
455         final PetriT generate_io_request = new PetriT("generate_io_request", 6.0);
456         generate_io_request.setDistribution("unif", generate_io_request.getTimeServ());
457         generate_io_request.setParamDeviation(5.33);
458         d_T.add(generate_io_request);
459         d_In.add(new ArcIn(generator_io_request, generate_io_request, 1));
460         d_Out.add(new ArcOut(generate_io_request, generate_io_request, 1));

```

```

461     final PetriP generated_io_request = new PetriP("generated_io_request", 0);
462     d_P.add(generated_io_request);
463     d_Out.add(new ArcOut(generate_io_request, generated_io_request, 1));
464     return generated_io_request;
465 }
466
467 private static PetriP create_interrupt_generator(ArrayList<PetriP> d_P,
ArrayList<PetriT> d_T, ArrayList<ArcIn> d_In, ArrayList<ArcOut> d_Out) {
468     final PetriP generator_interrupt = new PetriP("generator_interrupt", 1);
469     d_P.add(generator_interrupt);
470     final PetriT generate_interrupt = new PetriT("generate_interrupt", 6.0);
471     generate_interrupt.setDistribution("exp", generate_interrupt.getTimeServ());
472     d_T.add(generate_interrupt);
473     d_In.add(new ArcIn(generator_interrupt, generate_interrupt, 1));
474     d_Out.add(new ArcOut(generate_interrupt, generate_interrupt, 1));
475     final PetriP generated_interrupt = new PetriP("generated_interrupt", 0);
476     d_P.add(generated_interrupt);
477     d_Out.add(new ArcOut(generate_interrupt, generated_interrupt, 1));
478     return generated_interrupt;
479 }
480 }
481
482
483 // ./src/main/java/ua/stetsenkoinna/course_work/GatherDataCourseWorkNet.java
484
485 package ua.stetsenkoinna.course_work;
486
487 import ua.stetsenkoinna.PetriObj.ExceptionInvalidTimeDelay;
488
489 import java.io.BufferedWriter;
490 import java.io.FileWriter;
491 import java.io.IOException;
492 import java.util.*;
493 import java.util.function.Consumer;

```

```
494
495 import java.io.File;
496 public class GatherDataCourseWorkNet {
497
498     static class DiffTimePoint {
499         public double diff;
500         public double timePoint;
501     }
502
503     static class PropertyStats {
504         List<Double> timePoint = new ArrayList<>();
505         List<Double> diskLoad = new ArrayList<>();
506         List<Double> ioChannelLoad = new ArrayList<>();
507         List<Double> processorsLoad = new ArrayList<>();
508         List<Double> useOfPage = new ArrayList<>();
509         List<Double> totalWaitAllocate = new ArrayList<>();
510         List<DiffTimePoint> timeInSystem = new ArrayList<>();
511         List<DiffTimePoint> timeWaitAllocate = new ArrayList<>();
512     }
513
514     static void sortDiffTimePointArray(final List<DiffTimePoint> diffTimePointArray) {
515         diffTimePointArray.sort(Comparator.comparingDouble(v -> v.timePoint));
516     }
517
518     static PropertyStats collectStats(
519         final double timeModelling,
520         final int pagesNum,
521         final int processorsNum,
522         final int diskNum,
523         final int pagesStart,
524         final int pagesEnd,
525         final double tasksTimeMean
526     ) {
527         final CourseWorkNet courseWorkNet;
```

```

528     try {
529         courseWorkNet = new CourseWorkNet(pagesNum, processorsNum, diskNum,
pagesStart, pagesEnd, tasksTimeMean);
530     } catch (ExceptionInvalidTimeDelay e) {
531         throw new RuntimeException(e);
532     }
533
534     final CourseWorkPetriSim sim = new CourseWorkPetriSim(courseWorkNet.net);
535
536     final PropertyStats propertyStats = new PropertyStats();
537
538     Consumer<Double> trackStats = (currentTimeModelling) -> {
539         propertyStats.timePoint.add(currentTimeModelling);
540         propertyStats.useOfPage.add((double)(pagesNum -
courseWorkNet.pages.getMark()));
541         propertyStats.totalWaitAllocate.add((double)
(courseWorkNet.total_wait_allocate_task.getMark()));
542
543         double totalPlaceDiskWorkTime = 0;
544         double totalIoChannelWorkTime = 0;
545         double totalProcessorsWorkTime = 0;
546
547         for (final CourseWorkNet.TaskObject taskObject : courseWorkNet.taskObjects) {
548             totalPlaceDiskWorkTime += taskObject.place_disk.getTotalTimeServ();
549             totalIoChannelWorkTime += taskObject.io_channel_transfer.getTotalTimeServ();
550             totalProcessorsWorkTime += taskObject.process.getTotalTimeServ();
551         }
552
553         final double diskLoad = currentTimeModelling < 0.000000001 ? 0 :
((totalPlaceDiskWorkTime / currentTimeModelling) / diskNum);
554         final double ioChannelLoad = currentTimeModelling < 0.000000001 ? 0 :
(totalIoChannelWorkTime / currentTimeModelling);
555         final double processorsLoad = currentTimeModelling < 0.000000001 ? 0 :
((totalProcessorsWorkTime / currentTimeModelling) / processorsNum);

```

```
556
557     propertyStats.diskLoad.add(diskLoad);
558     propertyStats.ioChannelLoad.add(ioChannelLoad);
559     propertyStats.processorsLoad.add(processorsLoad);
560 };
561
562     sim.go(timeModelling, trackStats);
563
564     for(final CourseWorkNet.TaskObject taskObject : courseWorkNet.taskObjects) {
565         updateDiffTimePointArray(
566             taskObject.io_channel_transfer.getOutMoments(),
567             taskObject.generate.getOutMoments(),
568             propertyStats.timeInSystem
569         );
570         updateDiffTimePointArray(
571             taskObject.wait_allocate.getOutMoments(),
572             taskObject.fail_allocate.getOutMoments(),
573             propertyStats.timeWaitAllocate
574         );
575     }
576     sortDiffTimePointArray(propertyStats.timeInSystem);
577     sortDiffTimePointArray(propertyStats.timeWaitAllocate);
578
579     return propertyStats;
580 }
581
582 static void writeStats(
583     final String writerCommonPropsName,
584     final String writerTimeInSystemName,
585     final String writerTimeWaitAllocateName,
586     final int totalRuns,
587     final double timeModelling,
588     final int pagesNum,
589     final int processorsNum,
```

```

590     final int diskNum,
591     final int pagesStart,
592     final int pagesEnd,
593     final double tasksTimeMean
594 ) throws IOException {
595     try (
596         BufferedWriter writerCommonProps = new BufferedWriter(new
FileWriter(writerCommonPropsName));
597         BufferedWriter writerTimeInSystem = new BufferedWriter(new
FileWriter(writerTimeInSystemName));
598         BufferedWriter writerTimeWaitAllocate = new BufferedWriter(new
FileWriter(writerTimeWaitAllocateName))
599     ) {
600         writerCommonProps.write("runNumber"
601             + "," + "timePoint"
602             + "," + "diskLoad"
603             + "," + "ioChannelLoad"
604             + "," + "processorsLoad"
605             + "," + "totalWaitAllocate"
606             + "," + "useOfPage"
607         );
608         writerCommonProps.newLine();
609
610         writerTimeWaitAllocate.write("runNumber,timePoint,timeWaitAllocate");
611         writerTimeWaitAllocate.newLine();
612
613         writerTimeInSystem.write("runNumber,timePoint,timeInSystem");
614         writerTimeInSystem.newLine();
615
616         for(int runNumber = 0; runNumber < totalRuns; runNumber++) {
617             final PropertyStats propertyStats = collectStats(
618                 timeModelling,
619                 pagesNum,
620                 processorsNum,

```

```

621         diskNum,
622         pagesStart,
623         pagesEnd,
624         tasksTimeMean
625     );
626
627     for (int i = 0; i < propertyStats.timePoint.size(); i++) {
628         writerCommonProps.write(
629             String.format("%d", runNumber) + ", "
630             + String.format("%.10f", propertyStats.timePoint.get(i)) + ", "
631             + String.format("%.10f", propertyStats.diskLoad.get(i)) + ", "
632             + String.format("%.10f", propertyStats.ioChannelLoad.get(i)) + ", "
633             + String.format("%.10f", propertyStats.processorsLoad.get(i)) + ", "
634             + String.format("%.10f", propertyStats.totalWaitAllocate.get(i)) + ", "
635             + String.format("%.10f", propertyStats.useOfPage.get(i))
636         );
637         writerCommonProps.newLine();
638     }
639
640     String str = String.format("%d", runNumber) + ", "
641         + String.format("%.10f", 0.0) + ", "
642         + String.format("%.10f", 0.0);
643     if(propertyStats.timeInSystem.isEmpty()) {
644         writerTimeInSystem.write(
645             str
646         );
647         writerTimeInSystem.newLine();
648     } else {
649         for (final DiffTimePoint diffTimePoint : propertyStats.timeInSystem) {
650             writerTimeInSystem.write(
651                 String.format("%d", runNumber) + ", "
652                 + String.format("%.10f", diffTimePoint.timePoint) + ", "
653                 + String.format("%.10f", diffTimePoint.diff)
654             );

```

```

655         writerTimeInSystem.newLine();
656     }
657 }
658
659 if(propertyStats.timeWaitAllocate.isEmpty()) {
660     writerTimeWaitAllocate.write(
661         str
662     );
663     writerTimeWaitAllocate.newLine();
664 } else {
665     for (final DiffTimePoint diffTimePoint : propertyStats.timeWaitAllocate) {
666         writerTimeWaitAllocate.write(
667             String.format("%d", runNumber) + ","
668                 + String.format("%.10f", diffTimePoint.timePoint) + ","
669                 + String.format("%.10f", diffTimePoint.diff)
670         );
671         writerTimeWaitAllocate.newLine();
672     }
673 }
674 }
675 }
676 }
677
678 static class DetermineDistribution {
679     public static void main(String[] args) throws IOException {
680         writeStats(
681             "commonProps.csv",
682             "timeInSystem.csv",
683             "timeWaitAllocate.csv",
684         ,
685         0000,
686         1,
687
688

```

```
689 ,
690 ,
691
692     );
693     }
694 }
695
696 static void updateDiffTimePointArray(
697     final List<Double> toPoints,
698     final List<Double> fromPoints,
699     final List<DiffTimePoint> diffTimePoints
700 ) {
701     for(int i = 0; i < toPoints.size(); i++) {
702         final double point = toPoints.get(i);
703         final DiffTimePoint diffTimePoint = new DiffTimePoint();
704         diffTimePoint.timePoint = point;
705         diffTimePoint.diff = point - fromPoints.get(i);
706         diffTimePoints.add(diffTimePoint);
707     }
708 }
709
710 static class VerifyModel {
711
712     public static void main(String[] args) throws IOException {
713         final int[] pagesNums = new int[] { 131, 131, 200, 400, 700, 1000,
714         1000};
715         final int[] processorsNums = new int[] { 2, 2, 4, 5, 12, 40, 30};
716         final int[] diskNums = new int[] { 4, 4, 5, 11, 12, 8, 30};
717         final int[] pagesStarts = new int[] { 20, 20, 30, 70, 30, 60, 70};
718         final int[] pagesEnds = new int[] { 60, 60, 40, 100, 70, 100, 80};
719         final int[] tasksTimeMeans = new int[] { 5, 7, 8, 8, 8, 9, 15};
720
721         for (int i = 0; i < pagesEnds.length; i++) {
722             final int pagesNum = pagesNums[i];
```

```

722     final int processorsNum = processorsNums[i];
723     final int diskNum = diskNums[i];
724     final int pagesStart = pagesStarts[i];
725     final int pagesEnd = pagesEnds[i];
726     final int tasksTimeMean = tasksTimeMeans[i];
727
728     String dirName = String.format("%d_%d_%d_%d_%d_%d", pagesNum,
processorsNum, diskNum, pagesStart, pagesEnd, tasksTimeMean);
729     File directory = new File(dirName);
730     if(!directory.exists()) {
731         directory.mkdir();
732     }
733     dirName += "/";
734
735     writeStats(
736         dirName + "commonProps.csv",
737         dirName + "timeInSystem.csv",
738         dirName + "timeWaitAllocate.csv",
739     5,
740     800000,
741         pagesNum,
742         processorsNum,
743         diskNum,
744         pagesStart,
745         pagesEnd,
746         tasksTimeMean
747     );
748 }
749 }
750 }
751
752 // static <T extends Number> double calculateAverage(
753 // final Stream<Double> timePointList,
754 // final Stream<T> values

```

```
755 // ) {
756 //     return calculateAverage(timePointList.mapToDouble(Double::doubleValue),
values.mapToDouble(T::doubleValue));
757 // }
758 //
759 // static double calculateAverage(final List<DiffTimePoint> list) {
760 //     return calculateAverage(
761 //         list.stream().map(v -> v.timePoint),
762 //         list.stream().map(v -> v.diff)
763 //     );
764 // }
765 //
766 // static double calculateAverage(
767 //     final DoubleStream timePointList,
768 //     final DoubleStream values
769 // ) {
770 //     final Iterator<Double> timePointIt = timePointList.iterator();
771 //     final Iterator<Double> valuesIt = values.iterator();
772 //
773 //     double prevTimePoint = timePointIt.next();
774 //     double delaySum = 0.0;
775 //     double valueSum = 0;
776 //
777 //     while(timePointIt.hasNext()) {
778 //         final double timePoint = timePointIt.next();
779 //         final double value = valuesIt.next();
780 //         final double delay = timePoint - prevTimePoint;
781 //         prevTimePoint = timePoint;
782 //         delaySum += delay;
783 //         valueSum += value * delay;
784 //     }
785 //
786 //     return valueSum / delaySum;
787 // }
```

```
788 //
789 //  static <T extends Number> double calculateStdDev(
790 //      final Stream<Double> timePointList,
791 //      final Stream<T> values,
792 //      final double mean
793 //  ){
794 //      return calculateStdDev(timePointList.mapToDouble(Double::doubleValue),
values.mapToDouble(T::doubleValue), mean);
795 //  }
796
797 //  static double calculateStdDev(final List<DiffTimePoint> list, final double mean) {
798 //      return calculateStdDev(
799 //          list.stream().map(v -> v.timePoint),
800 //          list.stream().map(v -> v.diff),
801 //          mean
802 //      );
803 //  }
804 //
805 //  static <T extends Number> double calculateStdDev(
806 //      final DoubleStream timePointList,
807 //      final DoubleStream values,
808 //      final double mean
809 //  ){
810 //      final Iterator<Double> timePointIt = timePointList.iterator();
811 //      final Iterator<Double> valuesIt = values.iterator();
812 //
813 //      double prevTimePoint = timePointIt.next();
814 //      double delaySum = 0.0;
815 //      double valueSum = 0;
816 //
817 //      while(timePointIt.hasNext()) {
818 //          final double timePoint = timePointIt.next();
819 //          final double value = valuesIt.next();
820 //          final double delay = timePoint - prevTimePoint;
```

```
821 //      delaySum += delay;
822 //      prevTimePoint = timePoint;
823 //      valueSum += Math.pow(value - mean, 2) * delay;
824 //  }
825 //
826 //      return Math.sqrt(valueSum / delaySum);
827 //  }
828 }
829
830 // ./src/main/java/ua/stetsenkoinna/PetriObj/ExceptionInvalidTimeDelay.java
831
832 /*
833  * To change this license header, choose License Headers in Project Properties.
834  * To change this template file, choose Tools | Templates
835  * and open the template in the editor.
836  */
837 package ua.stetsenkoinna.PetriObj;
838
839 /**
840  * This exception is generated when user tries to construct Petri net with transition
841  * that has no input positions or output positions,
842  * and also if Petri net has no any position or any transition.
843  * @author Inna V. Stetsenko
844  */
845 public class ExceptionInvalidTimeDelay extends Exception {
846
847     public ExceptionInvalidTimeDelay(String string) {
848         super(string);
849     }
850
851 }
852
853
854 // ./src/main/java/ua/stetsenkoinna/PetriObj/ArcIn.java
```

```
855
856 package ua.stetsenkoinna.PetriObj;
857
858 public class ArcIn {
859
860     private final int numP;
861     private final int numT;
862     private final int k;
863     boolean inf;
864
865     public ArcIn(PetriP P, PetriT T) {
866         numP = P.getNumber();
867         numT = T.getNumber();
868         k = 1;
869         inf = false;
870     }
871
872     public ArcIn(PetriP P, PetriT T, int K) {
873         numP = P.getNumber();
874         numT = T.getNumber();
875         k = K;
876         inf = false;
877     }
878
879     public int getQuantity() {
880         return k;
881     }
882
883     public int getNumP() {
884         return numP;
885     }
886
887     public int getNumT() {
888         return numT;
```

```
889     }
890
891     public boolean getIsInf() {
892         return inf;
893     }
894 }
895
896
897 // ./src/main/java/ua/stetsenkoinna/PetriObj/ArcOut.java
898
899 package ua.stetsenkoinna.PetriObj;
900
901 public class ArcOut {
902
903     private final int numP;
904     private final int numT;
905     private final int k;
906
907     public ArcOut(PetriT T, PetriP P, int K) {
908         numP = P.getNumber();
909         numT = T.getNumber();
910         k = K;
911     }
912
913     public int getQuantity() {
914         return k;
915     }
916
917     public int getNumP() {
918         return numP;
919     }
920
921     public int getNumT() {
922         return numT;
```

```
923     }
924 }
925
926 // ./src/main/java/ua/stetsenkoinna/PetriObj/FunRand.java
927
928 package ua.stetsenkoinna.PetriObj;
929 import org.apache.commons.math3.distribution.PoissonDistribution;
930
931 import java.util.Random;
932
933 public class FunRand {
934
935     public static double exp(double timeMean) {
936         double a = 0;
937         while (a == 0) {
938             a = Math.random();
939         }
940         a = -timeMean * Math.log(a);
941
942         return a;
943     }
944
945     public static double unif(double timeMin, double timeMax) throws
ExceptionInvalidTimeDelay {
946         double a = 0;
947         while (a == 0) {
948             a = Math.random();
949         }
950         a = timeMin + a * (timeMax - timeMin);
951         if (a < 0)
952             throw new ExceptionInvalidTimeDelay("Negative time delay is generatated: Check
parameters for time delay.");
953         return a;
954     }
```

```
955
956  public static double norm(double timeMean, double timeDeviation) throws
ExceptionInvalidTimeDelay {
957      double a;
958      Random r = new Random();
959      a = timeMean + timeDeviation * r.nextGaussian();
960      if (a<0)
961          throw new ExceptionInvalidTimeDelay("Negative time delay is generatated: Check
parameters for time delay.");
962      return a;
963  }
964
965  public static double poisson(final double timeMean) {
966      PoissonDistribution poisson = new PoissonDistribution(timeMean);
967      return poisson.sample();
968  }
969 }
970
971
972 // ./src/main/java/ua/stetsenkoinna/PetriObj/PetriP.java
973
974 package ua.stetsenkoinna.PetriObj;
975
976 /*
977  * To change this template, choose Tools | Templates
978  * and open the template in the editor.
979  */
980 /**
981  * This class for creating the place of Petri net.
982  *
983  * @author Inna V. Stetsenko
984  */
985  public class PetriP {
986
```

```
987  private int mark;
988  private final String name;
989  private final int number;
990  private double mean;
991  private static int next = 0; //додано 1.10.2012, лічильник об'єктів
992  private int observedMax;
993  private int observedMin;
994
995  /**
996   *
997   * @param n name of place
998   * @param m quantity of markers
999   */
1000  public PetriP(String n, int m) {
1001      name = n;
1002      mark = m;
1003      mean = 0;
1004      number = next; //додано 1.10.2012
1005      next++;
1006      observedMax = m;
1007      observedMin = m;
1008  }
1009
1010  /**
1011   *
1012   * @param n - the name of place
1013   */
1014  public PetriP(String n) { //changed by Inna 21.03.2018
1015      this(n, 0);
1016
1017  }
1018
1019  public static void initNext() { //ініціалізація лічильника нульовим значенням
1020      next = 0;
```

```
1021     }
1022
1023     /**
1024      * /**
1025      * Recalculates the mean value
1026      *
1027      * @param a value for recalculate of mean value (value equals product of
1028      * marking and time divided by time modeling)
1029      */
1030     public void changeMean(double a) {
1031         mean = mean + (mark - mean) * a;
1032     }
1033
1034     /**
1035      *
1036      * @return mean value of quantity of markers
1037      */
1038     public double getMean() {
1039         return mean;
1040     }
1041
1042     /**
1043      *
1044      * @param a value on which increase the quantity of markers
1045      */
1046     public void increaseMark(int a) {
1047         mark += a;
1048         if (observedMax < mark) {
1049             observedMax = mark;
1050         }
1051         if (observedMin > mark) {
1052             observedMin = mark;
1053         }
1054
```

```
1055     }
1056
1057     /**
1058      *
1059      * @param a value on which decrease the quantity of markers
1060      */
1061     public void decreaseMark(int a) {
1062         mark -= a;
1063         if (observedMax < mark) {
1064             observedMax = mark;
1065         }
1066         if (observedMin > mark) {
1067             observedMin = mark;
1068         }
1069     }
1070
1071     /**
1072      *
1073      * @return current quantity of markers
1074      */
1075     public int getMark() {
1076         return mark;
1077     }
1078
1079     public String getName() {
1080         return name;
1081     }
1082
1083     /**
1084      *
1085      * @return number of the place
1086      */
1087     public int getNumber() {
1088         return number;
```

```
1089     }
1090
1091 }
1092
1093
1094 // ./src/main/java/ua/stetsenkoinna/PetriObj/StateTime.java
1095
1096 package ua.stetsenkoinna.PetriObj;
1097
1098 /**
1099  *
1100  * @author Anatoliy
1101  */
1102 public class StateTime {
1103     private double currentTime;
1104     private double simulationTime;
1105
1106     public StateTime() {
1107         currentTime = 0;
1108         simulationTime = Double.MAX_VALUE - 1;
1109     }
1110
1111     public double getCurrentTime() {
1112         return currentTime;
1113     }
1114
1115     public void setCurrentTime(double currentTime) {
1116         this.currentTime = currentTime;
1117     }
1118
1119     public double getSimulationTime() {
1120         return simulationTime;
1121     }
1122
```

```
1123  public void setSimulationTime(double modelingTime) {
1124      this.simulationTime = modelingTime;
1125  }
1126 }
1127
1128
1129 // ./src/main/java/ua/stetsenkoinna/PetriObj/PetriT.java
1130
1131 package ua.stetsenkoinna.PetriObj;
1132
1133 /*
1134  * To change this template, choose Tools | Templates
1135  * and open the template in the editor.
1136 */
1137 import java.util.*;
1138 import java.util.logging.Level;
1139 import java.util.logging.Logger;
1140
1141 /**
1142  * This class for creating the transition of Petri net
1143  *
1144  * @author Inna V. Stetsenko
1145 */
1146 public class PetriT {
1147
1148     private final String name;
1149     private int buffer;
1150     private int priority;
1151     private double probability;
1152
1153     private double minTime;
1154     private double timeServ;
1155     private double totalTimeServ;
1156     private double parametr; //середнє значення часу обслуговування
```

```

1157 private double paramDeviation; //середнє квадратичне відхилення часу обслуговування
1158 private String distribution;
1159
1160 private final ArrayList<Double> timeOut = new ArrayList<>();
1161 private final ArrayList<Integer> inP = new ArrayList<>();
1162 private final ArrayList<Integer> inPwithInf = new ArrayList<>();
1163 private final ArrayList<Integer> quantIn = new ArrayList<>();
1164 private final ArrayList<Integer> quantInwithInf = new ArrayList<>();
1165 private final ArrayList<Integer> outP = new ArrayList<>();
1166 private final ArrayList<Integer> quantOut = new ArrayList<>();
1167
1168 private int num; // номер каналу багатоканального переходу, що відповідає
найближчий події
1169 private final int number; // номер переходу за списком
1170 private static int next = 0; //додано 1.10.2012
1171
1172 private final ArrayList<Double> outMoments = new ArrayList<>();
1173 private boolean moments = false;
1174
1175 /**
1176  *
1177  * @param n name of transition
1178  * @param tS timed delay
1179  */
1180 public PetriT(String n, double tS) {
1181     name = n;
1182     parametr = tS;
1183     paramDeviation = 0;
1184     timeServ = parametr;
1185     buffer = 0;
1186
1187     minTime = Double.MAX_VALUE; // не очікується вихід маркерів переходу
1188     num = 0;
1189     priority = 0;

```

```

1190     probability = 1.0;
1191     distribution = null;
1192     number = next;
1193     next++;
1194     timeOut.add(Double.MAX_VALUE); // не очікується вихід маркерів з каналів
переходу
1195     this.minEvent();
1196 }
1197
1198
1199 public PetriT(String n) { //changed by Inna 21.03.2018
1200     this(n,0.0); //parametr = 0.0
1201 }
1202
1203 /**
1204  * Set the counter of transitions to zero.
1205  */
1206 public static void initNext() { //ініціалізація лічильника нульовим значенням
1207     next = 0;
1208 }
1209
1210 /**
1211  *
1212  * @return the value of priority
1213  */
1214 public int getPriority() {
1215     return priority;
1216 }
1217
1218 /**
1219  * Set the new value of priority
1220  *
1221  * @param r - the new value of priority
1222  */

```

```
1223  public void setPriority(int r) {
1224      priority = r;
1225  }
1226
1227  /**
1228   *
1229   * @return the value of priority
1230   */
1231  public double getProbability() {
1232      return probability;
1233  }
1234
1235  /**
1236   * Set the new value of probability
1237   *
1238   * @param v the value of probability
1239   */
1240  public void setProbability(double v) {
1241      probability = v;
1242  }
1243
1244  /**
1245   *
1246   * @return the numbers of planed moments of markers outputs
1247   */
1248  public int getBuffer() {
1249      return buffer;
1250  }
1251
1252  /**
1253   * This method sets the distribution of service time
1254   *
1255   * @param s the name of distribution as "exp", "norm", "unif". If s
1256   * equals null then the service time is determine value
```

```
1257  * @param param - the mean value of service time. If s equals null then the
1258  * service time equals <i>param</i>.
1259  */
1260  public void setDistribution(String s, double param) {
1261      distribution = s;
1262      parametr = param;
1263      timeServ = parametr; // додано 26.12.2011, моді, якщо s==null, то передається час
обслуговування
1264  }
1265
1266  /**
1267   *
1268   * @return current value of service time
1269   */
1270  public double getTimeServ() {
1271      double a = timeServ;
1272      if (distribution != null) {
1273          a = generateTimeServ();
1274      }
1275      return a;
1276  }
1277
1278  public double getTotalTimeServ() {
1279      return totalTimeServ;
1280  }
1281
1282  /**
1283   * Generating the value of service time
1284   *
1285   * @return value of service time which has been generated
1286   */
1287  public double generateTimeServ() {
1288      try {
1289          if (distribution != null) {
```

```

1290         if (distribution.equalsIgnoreCase("exp")) {
1291             timeServ = FunRand.exp(parametr);
1292         } else if (distribution.equalsIgnoreCase("unif")) {
1293             timeServ = FunRand.unif(parametr - paramDeviation, parametr +
paramDeviation);
1294         } else if (distribution.equalsIgnoreCase("norm")) {
1295             timeServ = FunRand.norm(parametr, paramDeviation);
1296         } else if (distribution.equalsIgnoreCase("poisson")) {
1297             timeServ = FunRand.poisson(parametr);
1298         }
1299     } else {
1300         timeServ = parametr;
1301     }
1302 } catch (ExceptionInvalidTimeDelay ex) {
1303     Logger.getLogger(PetriT.class.getName()).log(Level.SEVERE, null, ex);
1304 }
1305 totalTimeServ += timeServ;
1306 return timeServ;
1307 }
1308
1309 /**
1310  *
1311  * @return the name of transition
1312  */
1313 public String getName() {
1314     return name;
1315 }
1316
1317 /**
1318  *
1319  * @return the time of nearest event
1320  */
1321 public double getMinTime() {
1322     this.minEvent();

```

```

1323     return minTime;
1324 }
1325
1326 /**
1327  *
1328  * @return the number of transition
1329  */
1330 public int getNumber() {
1331     return number;
1332 }
1333
1334 /**
1335  * This method determines the places which is input for the transition. <br>
1336  * The class PetriNet use this method for creating net with given arrays of
1337  * places, transitions, input arcs and output arcs.
1338  *
1339  * @param inPP array of places // не використовується методом, видалити
1340  * @param arcs array of input arcs
1341  * @throws ExceptionInvalidTimeDelay if Petri net has invalid structure
1342  */
1343 public void createInP(ArcIn[] arcs) throws ExceptionInvalidTimeDelay {
1344     inPwithInf.clear(); //додано 28.11.2012 список має бути порожнім!!!
1345     quantInwithInf.clear(); //додано 28.11.2012
1346     inP.clear(); //додано 28.11.2012
1347     quantIn.clear(); //додано 28.11.2012
1348     for (ArcIn arc: arcs) {
1349         if (arc.getNumT() == this.getNumber()) {
1350             if (arc.getIsInf()) {
1351                 inPwithInf.add(arc.getNumP());
1352                 quantInwithInf.add(arc.getQuantity());
1353             } else {
1354                 //if (arcs[j].getQuantity() > 0) { //вхідна позиція додається у разі позитивної
1355                 //кількості зв'язків, 9.11.2015
1356                 inP.add(arc.getNumP());

```

```

1356         quantIn.add(arc.getQuantity());
1357         // }
1358     }
1359 }
1360 }
1361 if (inP.isEmpty()) {
1362     throw new ExceptionInvalidTimeDelay("Transition " + this.getName() + " hasn't
input positions!");
1363 }
1364
1365 }
1366
1367 /**
1368  * This method determines the places which is output for the transition.
1369  * <br>
1370  * The class PetriNet use this method for creating net with given arrays of
1371  * places, transitions, input arcs and output arcs.
1372  *
1373  * @param inPP array of places
1374  * @param arcs array of output arcs
1375  * @throws ExceptionInvalidTimeDelay if Petri net has invalid structure
1376  */
1377 public void createOutP(ArcOut[] arcs) throws ExceptionInvalidTimeDelay {
1378     getOutP().clear(); //додано 28.11.2012
1379     quantOut.clear(); //додано 28.11.2012
1380     for (ArcOut arc: arcs) {
1381         if ( arc.getNumT() == this.getNumber()) {
1382             getOutP().add(arc.getNumP());
1383             quantOut.add(arc.getQuantity());
1384         }
1385     }
1386     if (getOutP().isEmpty()) {
1387         throw new ExceptionInvalidTimeDelay("Transition " + this.getName() + " hasn't
output positions!");

```

```

1388     }
1389 }
1390
1391 /**
1392  * This method determines is firing condition of transition true.<br>
1393  * Condition is true if for each input place the quality of tokens in ....
1394  *
1395  * @param pp array of places of Petri net
1396  * @return true if firing condition is executed
1397  */
1398 public boolean condition(PetriP[] pp) { //Нумерація позицій тут відносна!!! inP.get(i)
- номер позиції у списку позицій, який побудований при конструюванні мережі Петрі,
1399
1400     boolean a = true;
1401     boolean b = true; // Саме тому при з'єднанні спільних позицій зміна номера не
призводить до трагічних наслідків (руйнування зв'язків)!!!
1402     for (int i = 0; i < inP.size(); i++) {
1403         if (pp[inP.get(i)].getMark() < quantIn.get(i)) {
1404             a = false;
1405             break;
1406         }
1407     }
1408     for (int i = 0; i < inPwithInf.size(); i++) {
1409         if (pp[inPwithInf.get(i)].getMark() < quantInwithInf.get(i)) {
1410             b = false;
1411             break;
1412         }
1413     }
1414     return a && b;
1415
1416 }
1417
1418 /**
1419  * The firing transition consists of two actions - tokens input and

```

```

1420  * output.<br>
1421  * This method provides tokens input in the transition.
1422  *
1423  * @param pp array of Petri net places
1424  * @param currentTime current time
1425  */
1426  public void actIn(PetriP[] pp, double currentTime) {
1427      if (this.condition(pp)) {
1428          for (int i = 0; i < inP.size(); i++) {
1429              pp[inP.get(i)].decreaseMark(quantIn.get(i));
1430          }
1431          if (buffer == 0) {
1432              timeOut.set(0, currentTime + this.getTimeServ());
1433          } else {
1434              timeOut.add(currentTime + this.getTimeServ());
1435          }
1436          buffer++;
1437          this.minEvent();
1438
1439      }
1440  }
1441
1442  /**
1443   * The firing transition consists of two actions - tokens input and
1444   * output.<br>
1445   * This method provides tokens output in the transition.
1446   *
1447   * @param pp array of Petri net places
1448   * @param currentTime current time
1449   */
1450  public void actOut(PetriP[] pp, double currentTime) { // parameter current time ia added
by Inna 11.07.2018 for protocol events
1451      if (buffer > 0) {
1452          for (int j = 0; j < getOutP().size(); j++) {

```

```

1453         pp[getOutP().get(j)].increaseMark(quantOut.get(j));
1454     }
1455     if (num == 0 && (timeOut.size() == 1)) {
1456         timeOut.set(0, Double.MAX_VALUE);
1457     } else {
1458         timeOut.remove(num);
1459     }
1460     if(moments){
1461         outMoments.add(currentTime);
1462     }
1463     buffer--;
1464 }
1465 }
1466
1467
1468 /**
1469  * Determines the transition nearest event among the events of its tokens
1470  * outputs. and the number of transition channel
1471  */
1472 public final void minEvent() {
1473     minTime = Double.MAX_VALUE;
1474     if (!timeOut.isEmpty()) {
1475         for (int i = 0; i < timeOut.size(); i++) {
1476             if (timeOut.get(i) < minTime) {
1477                 minTime = timeOut.get(i);
1478                 num = i;
1479             }
1480         }
1481     }
1482
1483 }
1484
1485 public ArrayList<Integer> getInP() {
1486     return inP;

```

```
1487     }
1488
1489     /**
1490     *
1491     * @return list of transition output places
1492     */
1493     public ArrayList<Integer> getOutP() {
1494         return outP;
1495     }
1496
1497     public void setParamDeviation(double parameter) {
1498         paramDeviation = parameter;
1499     }
1500
1501     /**
1502     * @return the outMoments
1503     */
1504     public ArrayList<Double> getOutMoments() {
1505         return outMoments;
1506     }
1507
1508     public void setMoments(boolean moments) {
1509         this.moments = moments;
1510     }
1511 }
1512
1513
1514 // ./src/main/java/ua/stetsenkoinna/PetriObj/PetriNet.java
1515
1516 package ua.stetsenkoinna.PetriObj;
1517
1518 import java.util.ArrayList;
1519
1520 /**
```

```

1521  * To change this template, choose Tools | Templates
1522  * and open the template in the editor.
1523  */
1524  /**
1525   * This class provides constructing Petri net
1526   *
1527   * @author Inna V. Stetsenko
1528   */
1529  public class PetriNet {
1530
1531      private final PetriP[] ListP;
1532      private final PetriT[] ListT;
1533
1534      /**
1535       * Construct Petri net for given set of places, set of transitions, set of
1536       * arcs and the name of Petri net
1537       *
1538       * @param s name of Petri net
1539       * @param pp set of places
1540       * @param TT set of transitions
1541       * @param In set of arcs directed from place to transition
1542       * @param Out set of arcs directed from transition to place
1543       */
1544
1545      public PetriNet(String s, ArrayList<PetriP> pp, ArrayList<PetriT> TT, ArrayList<ArcIn>
In, ArrayList<ArcOut> Out) throws ExceptionInvalidTimeDelay //додано 16 серпня 2011
1546      { //Працює прекрасно, якщо номери у списку співпадають із номерами, що
присвоюються, і з номерами, які використовувались при створенні зв'язків!!
1547          int numP = pp.size();
1548          int numT = TT.size();
1549          int numIn = In.size();
1550          int numOut = Out.size();
1551          ListP = new PetriP[numP];
1552          ListT = new PetriT[numT];

```

```

1553     ArcIn[] listIn = new ArcIn[numIn];
1554     ArcOut[] listOut = new ArcOut[numOut];
1555
1556     for (int j = 0; j < numP; j++) {
1557         ListP[j] = pp.get(j);
1558     }
1559
1560     for (int j = 0; j < numT; j++) {
1561         ListT[j] = TT.get(j);
1562     }
1563
1564     for (int j = 0; j < numIn; j++) {
1565         listIn[j] = In.get(j);
1566     }
1567     for (int j = 0; j < numOut; j++) {
1568         listOut[j] = Out.get(j);
1569     }
1570
1571     for (PetriT transition : ListT) {
1572         transition.createInP(listIn);
1573         transition.createOutP(listOut);
1574     }
1575
1576 }
1577
1578 /**
1579  *
1580  * @return array of Petri net places
1581  */
1582 public PetriP[] getListP() {
1583     return ListP;
1584 }
1585
1586 /**

```

```
1587      *
1588      * @return array of Petri net transitions
1589      */
1590      public PetriT[] getListT() {
1591          return ListT;
1592      }
1593 }
1594
1595
1596 // ./src/main/java/ua/stetsenkoinna/PetriObj/ExceptionInvalidNetStructure.java
1597
1598 /*
1599  * To change this license header, choose License Headers in Project Properties.
1600  * To change this template file, choose Tools | Templates
1601  * and open the template in the editor.
1602  */
1603 package ua.stetsenkoinna.PetriObj;
1604
1605 /**
1606  *
1607  * @author Inna V. Stetsenko
1608  */
1609 public class ExceptionInvalidNetStructure extends Exception {
1610
1611      public ExceptionInvalidNetStructure(String string) {
1612          super(string);
1613      }
1614
1615
1616 }
1617
1618
1619
1620 1 #!/usr/bin/env python
```

```
2 # coding: utf-8
3
4 # In[21]:
5
6
7 import numpy as np
8 import pandas as pd
9
10 def calculate_mean(time_points: pd.Series, values: pd.Series) -> float:
11     prev_time_point = time_points.iloc[0]
12     delay_sum = 0.0
13     value_sum = 0.0
14
15     for time_point, value in zip(time_points.iloc[1:], values):
16         delay = time_point - prev_time_point
17         prev_time_point = time_point
18         delay_sum += delay
19         value_sum += value * delay
20
21     try:
22         res = value_sum / delay_sum
23         return res
24     except ZeroDivisionError:
25         return 0
26
27 def calculate_std_dev(time_points: pd.Series, values: pd.Series, mean: float) -> float:
28     prev_time_point = time_points.iloc[0]
29     delay_sum = 0.0
30     value_sum = 0.0
31
32     for time_point, value in zip(time_points.iloc[1:], values):
33         delay = time_point - prev_time_point
34         prev_time_point = time_point
35         delay_sum += delay
```

```
36     value_sum += ((value - mean) ** 2) * delay
37
38     try:
39         res = np.sqrt(value_sum / delay_sum)
40         return res
41     except ZeroDivisionError:
42         return 0
43
44
45 # In[22]:
46
47
48 import os
49 from pathlib import Path
50 import pandas as pd
51 import attr
52 from typing import Optional
53
54 verification_data_dir = Path('./verification_data')
55
56 @attr.frozen
57 class ParamsData:
58     common_props: pd.DataFrame
59     time_wait_allocate: pd.DataFrame
60     time_in_system: pd.DataFrame
61
62 datas: list[ParamsData] = []
63 params_mat: list[pd.Series] = []
64
65 for dirpath, dir, filenames in os.walk(verification_data_dir):
66     dir_path = Path(dirpath)
67     if dir_path.name == verification_data_dir.name:
68         continue
69     params = tuple(int(n) for n in dir_path.name.split('_'))
```

```

70
71 params_mat.append(pd.Series({
72     'Кількість сторінок': params[0],
73     'Кількість процесорів': params[1],
74     'Кількість дисків': params[2],
75     'Початок сторінок': params[3],
76     'Кінець сторінок': params[4],
77     'Середій інтервал надходження завдань': params[5]
78 })))
79
80 common_props: Optional[pd.DataFrame] = None
81 time_wait_allocate: Optional[pd.DataFrame] = None
82 time_in_system: Optional[pd.DataFrame] = None
83 for file_name in filenames:
84     data = pd.read_csv(Path(dirpath) / file_name)
85     if file_name.startswith('commonProps'):
86         common_props = data
87         # threshold = 0.01
88         # common_props['processorsLoad'] = common_props['processorsLoad'].apply(lambda
x: 0 if x < threshold else x)
89         # common_props['diskLoad'] = common_props['diskLoad'].apply(lambda x: 0 if x <
threshold else x)
90         # common_props['ioChannelLoad'] = common_props['ioChannelLoad'].apply(lambda
x: 0 if x < threshold else x)
91     elif file_name.startswith('timeWaitAllocate'):
92         time_wait_allocate = data
93     elif file_name.startswith('timeInSystem'):
94         time_in_system = data
95
96     if common_props is not None and time_wait_allocate is not None and time_in_system is
not None:
97         datas.append(ParamsData(common_props, time_wait_allocate, time_in_system))
98     else:
99         raise Exception('empty data')

```

```
100
101
102 # In[23]:
103
104
105 params_data_frame = pd.concat(params_mat, axis=1)
106 params_data_frame = params_data_frame.T
107 params_data_frame.to_csv(verification_res_dir_path / 'params.csv', index=True,
index_label='Индекс')
108 params_data_frame
109
110
111 # In[24]:
112
113
114 from array import array
115
116 @attr.frozen
117 class MeanStddevStats:
118     diskLoad_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
119     diskLoad_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
120     ioChannelLoad_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
121     ioChannelLoad_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
122     processorsLoad_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
123     processorsLoad_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
124     totalWaitAllocate_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
125     totalWaitAllocate_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
126     useOfPage_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
127     useOfPage_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
128     timeInSystem_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
129     timeInSystem_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
130     timeWaitAllocate_mean: array[float] = attr.field(init=False, factory=lambda: array('d'))
131     timeWaitAllocate_std_dev: array[float] = attr.field(init=False, factory=lambda: array('d'))
132
```

```

133 mean_stddev_stats_list: list[pd.DataFrame] = []
134
135 for index, params_data in enumerate(datas):
136     mean_stddev_stats = MeanStddevStats()
137
138     for run_num, group in params_data.common_props.groupby('runNumber'):
139         # Calculate means and standard deviations
140         diskLoad_mean = calculate_mean(group['timePoint'], group['diskLoad'])
141         diskLoad_std_dev = calculate_std_dev(group['timePoint'], group['diskLoad'],
diskLoad_mean)
142
143         ioChannelLoad_mean = calculate_mean(group['timePoint'], group['ioChannelLoad'])
144         ioChannelLoad_std_dev = calculate_std_dev(group['timePoint'],
group['ioChannelLoad'], ioChannelLoad_mean)
145
146         processorsLoad_mean = calculate_mean(group['timePoint'], group['processorsLoad'])
147         processorsLoad_std_dev = calculate_std_dev(group['timePoint'],
group['processorsLoad'], processorsLoad_mean)
148
149         totalWaitAllocate_mean = calculate_mean(group['timePoint'], group['totalWaitAllocate'])
150         totalWaitAllocate_std_dev = calculate_std_dev(group['timePoint'],
group['totalWaitAllocate'], totalWaitAllocate_mean)
151
152         useOfPage_mean = calculate_mean(group['timePoint'], group['useOfPage'])
153         useOfPage_std_dev = calculate_std_dev(group['timePoint'], group['useOfPage'],
useOfPage_mean)
154
155         mean_stddev_stats.diskLoad_mean.append(diskLoad_mean)
156         mean_stddev_stats.diskLoad_std_dev.append(diskLoad_std_dev)
157
158         mean_stddev_stats.ioChannelLoad_mean.append(ioChannelLoad_mean)
159         mean_stddev_stats.ioChannelLoad_std_dev.append(ioChannelLoad_std_dev)
160
161         mean_stddev_stats.processorsLoad_mean.append(processorsLoad_mean)

```

```

162     mean_stddev_stats.processorsLoad_std_dev.append(processorsLoad_std_dev)
163
164     mean_stddev_stats.totalWaitAllocate_mean.append(totalWaitAllocate_mean)
165     mean_stddev_stats.totalWaitAllocate_std_dev.append(totalWaitAllocate_std_dev)
166
167     mean_stddev_stats.useOfPage_mean.append(useOfPage_mean)
168     mean_stddev_stats.useOfPage_std_dev.append(useOfPage_std_dev)
169
170     for run_num, group in params_data.time_in_system.groupby('runNumber'):
171         timeInSystem_mean = calculate_mean(group['timePoint'], group['timeInSystem'])
172         timeInSystem_std_dev = calculate_std_dev(group['timePoint'], group['timeInSystem'],
timeInSystem_mean)
173         mean_stddev_stats.timeInSystem_mean.append(timeInSystem_mean)
174         mean_stddev_stats.timeInSystem_std_dev.append(timeInSystem_std_dev)
175
176     for run_num, group in params_data.time_wait_allocate.groupby('runNumber'):
177         timeWaitAllocate_mean = calculate_mean(group['timePoint'], group['timeWaitAllocate'])
178         timeWaitAllocate_std_dev = calculate_std_dev(group['timePoint'],
group['timeWaitAllocate'], timeWaitAllocate_mean)
179         mean_stddev_stats.timeWaitAllocate_mean.append(timeWaitAllocate_mean)
180         mean_stddev_stats.timeWaitAllocate_std_dev.append(timeWaitAllocate_std_dev)
181
182     dt = pd.DataFrame(attr.asdict(mean_stddev_stats))
183     dt['params_index'] = index
184     mean_stddev_stats_list.append(dt)
185
186
187 # In[48]:
188
189
190     rename_dict = {
191         'diskLoad': 'Завантаження дисків',
192         'ioChannelLoad': 'Завантаження каналу введення-виведення',
193         'processorsLoad': 'Завантаження процесорів',

```

```
194     'totalWaitAllocate': "Кількість завдань в очікуванні пам'яті",
195     'useOfPage': 'Кількість зайнятих сторінок',
196     'timeInSystem': 'Час завдання в системі',
197     'timeWaitAllocate': "Час виділення пам'яті",
198 }
199
200
201 def split_into_means_and_stddevs(data: pd.DataFrame) -> tuple[pd.DataFrame,
pd.DataFrame]:
202     means = pd.DataFrame()
203     stddevs = pd.DataFrame()
204     for name in data.columns:
205         short_name = name.split('_')[0]
206         if name.endswith('mean'):
207             means[short_name] = data[name]
208         else:
209             stddevs[short_name] = data[name]
210
211     means['Індекс набору параметрів'] = data['params_index']
212     stddevs['Індекс набору параметрів'] = data['params_index']
213     means.rename(columns=rename_dict, inplace=True)
214     stddevs.rename(columns=rename_dict, inplace=True)
215
216     column_names = means.columns.tolist()
217     column_names = [column_names[-1]] + column_names[:-1]
218
219     means = means[column_names]
220     stddevs = stddevs[column_names]
221
222     return means, stddevs
223
224
225 # In[49]:
226
```

```
227
228 mean_stddev_stats_data_frame = pd.concat(mean_stddev_stats_list, ignore_index=True)
229 mean_stats_data_frame, stddev_stats_data_frame =
split_into_means_and_stddevs(mean_stddev_stats_data_frame)
230
231
232 # In[50]:
233
234
235 mean_stats_data_frame.to_csv(verification_res_dir_path / 'mean_stats_data_frame.csv',
index=False)
236 mean_stats_data_frame
237
238
239 # In[51]:
240
241
242 stddev_stats_data_frame.to_csv(verification_res_dir_path / 'stddev_stats_data_frame.csv',
index=False)
243 stddev_stats_data_frame
244
245
246 # In[52]:
247
248
249 global_mean_stddev_list: list[pd.DataFrame] = []
250 mean_stddev_stats_relative_mean_list: list[pd.DataFrame] = []
251
252 for i, mean_stddev_stats in mean_stddev_stats_data_frame.groupby('params_index'):
253     means = mean_stddev_stats.mean()
254     global_mean_stddev_list.append(means)
255     mean_stddev_stats_relative_mean = ((mean_stddev_stats - means).abs() * 100) / means
256     mean_stddev_stats_relative_mean.fillna(0, inplace=True)
257     mean_stddev_stats_relative_mean['params_index'] = i
```

```
258 mean_stddev_stats_relative_mean_list.append(mean_stddev_stats_relative_mean)
259
260
261 # In[53]:
262
263
264 global_mean_data_frame = pd.DataFrame()
265 global_std_dev_data_frame = pd.DataFrame()
266 for name in global_mean_stddev_data_frame.columns:
267     short_name = name.split('_')[0]
268     if name.endswith('mean'):
269         global_mean_data_frame[short_name] = global_mean_stddev_data_frame[name]
270     else:
271         global_std_dev_data_frame[short_name] = global_mean_stddev_data_frame[name]
272
273 global_mean_data_frame.rename(columns=rename_dict, inplace=True)
274 global_std_dev_data_frame.rename(columns=rename_dict, inplace=True)
275
276
277 # In[54]:
278
279
280 global_mean_data_frame
281
282
283 # In[55]:
284
285
286 global_mean_data_frame.to_csv(verification_res_dir_path / 'global_mean_data_frame.csv',
index=True, index_label='Індекс набору параметрів')
287
288
289 # In[56]:
290
```

```
291
292 global_std_dev_data_frame
293
294
295 # In[57]:
296
297
298 global_std_dev_data_frame.to_csv(verification_res_dir_path /
'global_std_dev_data_frame.csv', index=True, index_label='Індекс набору параметрів')
299
300
301 # In[58]:
302
303
304 mean_stddev_stats_relative_mean_data_frame =
pd.concat(mean_stddev_stats_relative_mean_list, ignore_index=True)
305 mean_stats_relative_mean_data_frame, stddev_stats_relative_mean_data_frame =
split_into_means_and_stddevs(mean_stddev_stats_relative_mean_data_frame)
306
307
308 # In[59]:
309
310
311 mean_stats_relative_mean_data_frame.to_csv(verification_res_dir_path /
'mean_stats_relative_mean_data_frame.csv', index=False)
312 mean_stats_relative_mean_data_frame
313
314
315 # In[60]:
316
317
318 stddev_stats_relative_mean_data_frame.to_csv(verification_res_dir_path /
'stddev_stats_relative_mean_data_frame.csv', index=False)
319 stddev_stats_relative_mean_data_frame
```

```
1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[2]:
5
6
7 from pathlib import Path
8 import pandas as pd
9 import matplotlib.pyplot as plt
10
11 folder_path = Path('final_data')
12 common_props_raw: pd.DataFrame = pd.read_csv(folder_path / 'commonProps.csv')
13 time_in_system_raw: pd.DataFrame = pd.read_csv(folder_path / 'timeInSystem.csv')
14 time_wait_allocate_raw: pd.DataFrame = pd.read_csv(folder_path / 'timeWaitAllocate.csv')
15 common_props_raw.sort_values(by='timePoint', inplace=True)
16 time_in_system_raw.sort_values(by='timePoint', inplace=True)
17 time_wait_allocate_raw.sort_values(by='timePoint', inplace=True)
18
19
20 # In[3]:
21
22
23 import numpy as np
24 from array import array
25 from typing import Sequence
26
27 def calculate_means_through_time(
28     time_points: pd.Series,
29     values: pd.Series
30 ) -> array[float]:
31     prev_time_point = time_points.iloc[0]
32     delay_sum = 0.0
33     value_sum = 0.0
34
```

```

35 values_through_time = array('d')
36
37 for time_point, value in zip(time_points.iloc[1:], values):
38     delay = time_point - prev_time_point
39     prev_time_point = time_point
40     delay_sum += delay
41     value_sum += value * delay
42     values_through_time.append(value_sum / delay_sum)
43
44 return values_through_time
45
46 def calculate_stddevs_through_time(
47     time_points: pd.Series,
48     values: pd.Series,
49     means: array[float]
50 ) -> array[float]:
51     prev_time_point = time_points.iloc[0]
52     delay_sum = 0.0
53     value_sum = 0.0
54
55     stddevs_through_time: array[float] = array('d')
56
57     for time_point, value, mean in zip(time_points.iloc[1:], values, means):
58         delay = time_point - prev_time_point
59         prev_time_point = time_point
60         delay_sum += delay
61         value_sum += ((value - mean) ** 2) * delay
62         stddevs_through_time.append(np.sqrt(value_sum / delay_sum))
63
64     return stddevs_through_time
65
66
67 def calculate_mean(time_points: pd.Series, values: pd.Series) -> float:
68     prev_time_point = time_points.iloc[0]

```

```
69     delay_sum = 0.0
70     value_sum = 0.0
71
72     for time_point, value in zip(time_points.iloc[1:], values):
73         delay = time_point - prev_time_point
74         prev_time_point = time_point
75         delay_sum += delay
76         value_sum += value * delay
77
78     return value_sum / delay_sum
79
80 def calculate_std_dev(time_points: pd.Series, values: pd.Series, mean: float) -> float:
81     prev_time_point = time_points.iloc[0]
82     delay_sum = 0.0
83     value_sum = 0.0
84
85     for time_point, value in zip(time_points.iloc[1:], values):
86         delay = time_point - prev_time_point
87         prev_time_point = time_point
88         delay_sum += delay
89         value_sum += ((value - mean) ** 2) * delay
90
91     return np.sqrt(value_sum / delay_sum)
92
93
94 # In[4]:
95
96
97 from collections import deque
98 import attr
99
100 @attr.frozen
101 class PropertyMeanStdDev:
102     mean: array[float]
```

```

103     stdDev: array[float]
104
105 def calc_mean_stddev_through_time(time_points: pd.Series, props: pd.Series) ->
PropertyMeanStdDev:
106     means = calculate_means_through_time(time_points, props)
107     return PropertyMeanStdDev(
108         means,
109         calculate_stddevs_through_time(time_points, props, means)
110     )
111
112 time_points_mat: deque[Sequence[float]] = deque()
113 disk_load_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
114 io_channel_load_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
115 processors_load_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
116 use_of_page_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
117 total_wait_allocate_stddev_mat: deque[PropertyMeanStdDev] = deque()
118
119 time_in_system_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
120 time_wait_allocate_mean_stddev_mat: deque[PropertyMeanStdDev] = deque()
121 time_in_system_time_points_mat: deque[PropertyMeanStdDev] = deque()
122 time_wait_allocate_time_points_mat: deque[PropertyMeanStdDev] = deque()
123
124 for run_number, common_props_raw_indexed in
common_props_raw.groupby(by='runNumber'):
125     time_points = common_props_raw_indexed['timePoint'][:-1]
126     disk_load_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['diskLoad']))
127     io_channel_load_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['ioChannelLoad']))
128     processors_load_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['processorsLoad']))
129     use_of_page_mean_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['useOfPage']))

```

```

130     total_wait_allocate_stddev_mat.append(calc_mean_stddev_through_time(time_points,
common_props_raw_indexed['totalWaitAllocate']))
131     time_points_mat.append(time_points[:-1])
132
133     for run_number, time_in_system_raw_indexed in
time_in_system_raw.groupby(by='runNumber'):
134         time_in_system_time_points = time_in_system_raw_indexed['timePoint']
135
time_in_system_mean_stddev_mat.append(calc_mean_stddev_through_time(time_in_system_time
_points, time_in_system_raw_indexed['timeInSystem']))
136     time_in_system_time_points_mat.append(time_in_system_time_points.iloc[:-1])
137
138     for run_number, time_wait_allocate_raw_indexed in
time_wait_allocate_raw.groupby(by='runNumber'):
139         time_wait_allocate_time_points = time_wait_allocate_raw_indexed['timePoint']
140
time_wait_allocate_mean_stddev_mat.append(calc_mean_stddev_through_time(time_wait_allocat
e_time_points, time_wait_allocate_raw_indexed['timeWaitAllocate']))
141     time_wait_allocate_time_points_mat.append(time_wait_allocate_time_points.iloc[:-1])
142
143
144     # In[45]:
145
146
147     from typing import Iterable
148
149     def plot_matrix(time_points_mat: Iterable[float], value_mat: Iterable[float], ylabel: str):
150         for run_num, row in enumerate(zip(time_points_mat, value_mat)):
151             time_points_row = row[0]
152             value_row = row[1]
153             plt.plot(time_points_row, value_row, label=f'Run {run_num}')
154
155             plt.title(f'{ylabel} vs. TimePoint for Each Run')
156             plt.xlabel('TimePoint')

```

```

157 plt.ylabel(ylabel)
158 plt.legend(title='Run Number', loc='upper right')
159 plt.grid(True)
160 plt.savefig(fname=f'{ylabel}.svg', format='svg')
161 plt.show()
162
163 def plot_matrix_mean_std_dev(time_points_mat: Sequence[float], value_mat:
deque[PropertyMeanStdDev], ylabel: str):
164     plot_matrix(time_points_mat, (v.mean for v in value_mat), ylabel + 'Mean')
165     plot_matrix(time_points_mat, (v.stdDev for v in value_mat), ylabel + 'StdDev')
166
167 plot_matrix_mean_std_dev(time_points_mat, disk_load_mean_stddev_mat, 'diskLoad')
168 plot_matrix_mean_std_dev(time_points_mat, io_channel_load_mean_stddev_mat,
'ioChannelLoad')
169 plot_matrix_mean_std_dev(time_points_mat, processors_load_mean_stddev_mat,
'processorsLoad')
170 plot_matrix_mean_std_dev(time_points_mat, use_of_page_mean_stddev_mat, 'useOfPage')
171 plot_matrix_mean_std_dev(time_points_mat, total_wait_allocate_stddev_mat,
'totalWaitAllocate')
172 plot_matrix_mean_std_dev(time_in_system_time_points_mat,
time_in_system_mean_stddev_mat, 'timeInSystem')
173 plot_matrix_mean_std_dev(time_wait_allocate_time_points_mat,
time_wait_allocate_mean_stddev_mat, 'timeWaitAllocate')
174
175
176 # In[6]:
177
178
179 transit_period_start_at = 600000
180
181
182 # In[7]:
183
184

```

```

185 common_props_stable = common_props_raw[common_props_raw['timePoint'] >
transit_period_start_at]
186 time_in_system_stable = time_in_system_raw[time_in_system_raw['timePoint'] >
transit_period_start_at]
187 time_wait_allocate_stable = time_wait_allocate_raw[time_wait_allocate_raw['timePoint'] >
transit_period_start_at]
188
189
190 # In[8]:
191
192
193 data_stable_mean: dict[str, array[float]] = {}
194 data_stable_stddev: dict[str, array[float]] = {}
195
196 def append_last_cell(
197     dest_mean: dict[str, array[float]],
198     dest_stddev: dict[str, array[float]],
199     src: pd.DataFrame, col: str
200 ) -> None:
201     if not col in dest_mean:
202         dest_mean[col] = array('d')
203         dest_stddev[col] = array('d')
204     mean = calculate_mean(src['timePoint'], src[col])
205     dest_mean[col].append(mean)
206     dest_stddev[col].append(calculate_std_dev(src['timePoint'], src[col], mean))
207
208 for run_number, stable_run in common_props_stable.groupby('runNumber'):
209     append_last_cell(data_stable_mean, data_stable_stddev, stable_run, 'diskLoad')
210     append_last_cell(data_stable_mean, data_stable_stddev, stable_run, 'ioChannelLoad')
211     append_last_cell(data_stable_mean, data_stable_stddev, stable_run, 'processorsLoad')
212     append_last_cell(data_stable_mean, data_stable_stddev, stable_run, 'totalWaitAllocate')
213     append_last_cell(data_stable_mean, data_stable_stddev, stable_run, 'useOfPage')
214
215 for run_number, stable_run in time_in_system_stable.groupby('runNumber'):

```

```
216     append_last_cell(data_stable_mean, data_stable_stddev, stable_run, 'timeInSystem')
217
218 for run_number, stable_run in time_wait_allocate_stable.groupby('runNumber'):
219     append_last_cell(data_stable_mean, data_stable_stddev, stable_run, 'timeWaitAllocate')
220
221 data_stable_mean: pd.DataFrame = pd.DataFrame(data_stable_mean)
222 data_stable_stddev: pd.DataFrame = pd.DataFrame(data_stable_stddev)
223
224
225 # In[9]:
226
227
228 rename_dict = {
229     'diskLoad': 'Завантаження дисків',
230     'ioChannelLoad': 'Завантаження каналу введення-виведення',
231     'processorsLoad': 'Завантаження процесорів',
232     'totalWaitAllocate': 'Кількість завдань в очікуванні пам'яті',
233     'useOfPage': 'Кількість зайнятих сторінок',
234     'timeInSystem': 'Час завдання в системі',
235     'timeWaitAllocate': 'Час виділення пам'яті',
236 }
237
238
239 # In[10]:
240
241
242 ukr_data_stable_mean = data_stable_mean.rename(columns=rename_dict)
243 ukr_data_stable_stddev = data_stable_stddev.rename(columns=rename_dict)
244 ukr_data_stable_mean.to_csv('ukr_data_stable_mean.csv', index=True, index_label='Номер прогону')
245 ukr_data_stable_stddev.to_csv('ukr_data_stable_stddev.csv', index=True,
index_label='Номер прогону')
246
247
```

```
248 # In[27]:
249
250
251 global_mean: dict[str, float] = {}
252 global_stddev: dict[str, float] = {}
253
254 def update_global_mean_stddev(
255     dest_mean: dict[str, float],
256     dest_stddev: dict[str, float],
257     src: pd.DataFrame,
258     col: str
259 ) -> None:
260     mean = calculate_mean(src['timePoint'], src[col])
261     dest_mean[col] = mean
262     dest_stddev[col] = calculate_std_dev(src['timePoint'], src[col], mean)
263
264 for run_number, stable_run in common_props_stable.groupby('runNumber'):
265     update_global_mean_stddev(global_mean, global_stddev, stable_run, 'diskLoad')
266     update_global_mean_stddev(global_mean, global_stddev, stable_run, 'ioChannelLoad')
267     update_global_mean_stddev(global_mean, global_stddev, stable_run, 'processorsLoad')
268     update_global_mean_stddev(global_mean, global_stddev, stable_run, 'totalWaitAllocate')
269     update_global_mean_stddev(global_mean, global_stddev, stable_run, 'useOfPage')
270
271 for run_number, stable_run in time_in_system_stable.groupby('runNumber'):
272     update_global_mean_stddev(global_mean, global_stddev, stable_run, 'timeInSystem')
273
274 for run_number, stable_run in time_wait_allocate_stable.groupby('runNumber'):
275     update_global_mean_stddev(global_mean, global_stddev, stable_run, 'timeWaitAllocate')
276
277
278 # In[30]:
279
280
281 global_mean: pd.Series = pd.Series(global_mean)
```

```
282 global_stddev: pd.Series = pd.Series(global_stddev)
283
284
285 # In[42]:
286
287
288 global_mean_stddev = pd.concat([global_mean, global_stddev], axis=1)
289 global_mean_stddev.rename(columns={0: 'Середнє значення', 1: 'Середньоквадратичне
відхилення'}, inplace=True)
290 global_mean_stddev.rename(index=rename_dict, inplace=True)
291 global_mean_stddev.to_csv('global_mean_stddev.csv', index=True, index_label='Назва
вихідного параметра')
292 global_mean_stddev
293
294
295 # In[46]:
296
297
298 def show_hist(data, col_name: str) -> None:
299     plt.figure(figsize=(10, 6))
300     plt.hist(data[col_name], bins=100, edgecolor='k', alpha=0.7)
301     plt.title(f'Histogram of {col_name}')
302     plt.xlabel(col_name)
303     plt.ylabel('Frequency')
304     plt.savefig(fname=f'hist{col_name}.svg', format='svg')
305     plt.grid(True)
306
307
308 # In[47]:
309
310
311 show_hist(common_props_stable, 'diskLoad')
312 show_hist(common_props_stable, 'ioChannelLoad')
313 show_hist(common_props_stable, 'processorsLoad')
```

```
314 show_hist(common_props_stable, 'totalWaitAllocate')
315 show_hist(common_props_stable, 'useOfPage')
316 show_hist(time_in_system_stable, 'timeInSystem')
317 show_hist(time_wait_allocate_stable, 'timeWaitAllocate')
318
319
320 # In[56]:
321
322
323 time_in_system_stable_filtered = time_in_system_stable
324 time_in_system_stable_filtered_mean = calculate_mean(
325     time_in_system_stable_filtered['timePoint'],
326     time_in_system_stable_filtered['timeInSystem']
327 )
328 time_in_system_stable_filtered_std_dev = calculate_std_dev(
329     time_in_system_stable_filtered['timePoint'],
330     time_in_system_stable_filtered['timeInSystem'],
331     time_in_system_stable_filtered_mean
332 )
333 print(time_in_system_stable_filtered_std_dev)
334
335 plt.figure(figsize=(10, 6))
336 plt.hist(
337     time_in_system_stable_filtered['timeInSystem'],
338     bins=130,
339     edgecolor='k',
340     alpha=0.7,
341     label=f'mean = {time_in_system_stable_filtered_mean}\nstd_dev =
{time_in_system_stable_filtered_std_dev}'
342 )
343 plt.title(f'Histogram of timeInSystem')
344 plt.xlabel('timeInSystem')
345 plt.ylabel('Frequency')
346 plt.legend(title='Stats')
```

```
347 plt.grid(True)
348
349
350 # In[57]:
351
352
353 from typing import Sequence
354 import numpy as np
355 from scipy.stats import chi2
356
357 def chi_squared_normality_test(data: Sequence[float], mean: float, std_dev: float) -> bool:
358     # Calculate the number of bins using Sturges' rule
359     num_bins = int(np.ceil(1 + 3.322 * np.log10(len(data))))
360
361     # Create the histogram of the data
362     observed_counts, bin_edges = np.histogram(data, bins=num_bins)
363
364     # Calculate expected frequencies for a normal distribution
365     expected_counts = []
366     for i in range(len(bin_edges) - 1):
367         # Calculate the cumulative probability for the bin range
368         bin_prob = (
369             (1 / (std_dev * np.sqrt(2 * np.pi))) *
370             (np.exp(-0.5 * ((bin_edges[i + 1] - mean) / std_dev) ** 2) -
371              np.exp(-0.5 * ((bin_edges[i] - mean) / std_dev) ** 2))
372         )
373         expected_counts.append(bin_prob * len(data))
374
375     # Perform chi-squared statistic
376     expected_counts = np.array(expected_counts)
377     chi_squared_stat = np.sum((observed_counts - expected_counts) ** 2 / expected_counts)
378
379     # Degrees of freedom = (number of bins - 1 - number of estimated parameters)
380     degrees_of_freedom = num_bins - 1 - 2
```

```
381
382  # Find the critical value for the chi-squared distribution
383  critical_value = chi2.ppf(0.95, degrees_of_freedom)
384
385  # Check if the chi-squared statistic is within the critical range
386  return chi_squared_stat < critical_value
387
388
389 # In[58]:
390
391
392 time_in_system_stable_filtered_is_normal = chi_squared_normality_test(
393     time_in_system_stable_filtered['timeInSystem'],
394     time_in_system_stable_filtered_mean,
395     time_in_system_stable_filtered_std_dev
396 )
397 time_in_system_stable_filtered_is_normal
398
399
400 # In[64]:
401
402
403 from scipy.stats import norm
404
405 count, bins, _ = plt.hist(time_in_system_stable_filtered['timeInSystem'], bins=200,
density=True, alpha=0.6, label='Histogram')
406 x = np.linspace(bins[0], bins[-1], 1000)
407 pdf = norm.pdf(
408     x,
409     time_in_system_stable_filtered_mean,
410     time_in_system_stable_filtered_std_dev
411 )
412 plt.plot(x, pdf, 'r-', lw=2, label='Normal Distribution')
413 plt.title('Histogram with Normal Distribution Overlay')
```

```
414 plt.xlabel('Value')
415 plt.ylabel('Density')
416 plt.legend()
417 plt.grid(True)
418 plt.savefig(fname=f'histNormTimeInSystem.svg', format='svg')
419 plt.show()
420
421
422 # In[65]:
423
424
425 from scipy.stats import expon
426
427 time_wait_allocate_stable_filtered = time_wait_allocate_stable
428
429 time_wait_allocate_stable_filtered_mean = calculate_mean(
430     time_wait_allocate_stable_filtered['timePoint'],
431     time_wait_allocate_stable_filtered['timeWaitAllocate']
432 )
433 time_wait_allocate_stable_filtered_std_dev = calculate_std_dev(
434     time_wait_allocate_stable_filtered['timePoint'],
435     time_wait_allocate_stable_filtered['timeWaitAllocate'],
436     time_wait_allocate_stable_filtered_mean
437 )
438
439 count, bins, _ = plt.hist(
440     time_wait_allocate_stable_filtered['timeWaitAllocate'],
441     bins=70,
442     density=True,
443     alpha=0.6,
444     label=f'mean = {time_wait_allocate_stable_filtered_mean}'
445 )
446 plt.title('Histogram with Exponential Distribution Overlay')
447 plt.xlabel('Value')
```

```
448 plt.ylabel('Density')
449 plt.legend()
450 plt.grid(True)
451 plt.show()
452
453
454 # In[66]:
455
456
457 def chi_squared_exponential_test(data: Sequence[float], mean: float, alpha: float = 0.05) ->
bool:
458     # Sort data and calculate the number of bins
459     sorted_data = np.sort(data)
460     n = len(data)
461     bin_width = 2 / np.sqrt(n) # Rule of thumb for bin width in exponential distribution
462     bins = int(np.ceil(1 / bin_width))
463     # Create bin edges and calculate expected frequencies
464     max_data = max(sorted_data)
465     bin_edges = np.linspace(0, max_data, bins + 1)
466     observed_counts, _ = np.histogram(sorted_data, bins=bin_edges)
467     # Calculate expected frequencies for exponential distribution
468     expected_counts = np.diff(len(data) * (1 - np.exp(-bin_edges / mean)))
469     # Calculate chi-squared statistic
470     chi_squared_stat = np.sum((observed_counts - expected_counts)**2 / expected_counts)
471     # Degrees of freedom
472     degrees_of_freedom = bins - 1
473     # Compute the p-value
474     p_value = 1 - chi2.cdf(chi_squared_stat, df=degrees_of_freedom)
475     return p_value < alpha
476
477
478 # In[67]:
479
480
```

```
481 chi_squared_exponential_test(
482     time_wait_allocate_stable_filtered['timeWaitAllocate'],
483     time_wait_allocate_stable_filtered_mean,
484     05
485 )
486
487
488 # In[69]:
489
490
491 count, bins, _ = plt.hist(time_wait_allocate_stable_filtered['timeWaitAllocate'], bins=70,
density=True, alpha=0.6, label='Histogram')
492 x = np.linspace(bins[0], bins[-1], 1000)
493
494 pdf = expon.pdf(x, scale=time_wait_allocate_stable_filtered_mean)
495 plt.plot(x, pdf, 'r-', lw=2, label='Exponential Distribution')
496 plt.title('Histogram with Exponential Distribution Overlay')
497 plt.xlabel('Value')
498 plt.ylabel('Density')
499 plt.legend()
500 plt.grid(True)
501 plt.savefig(fname=f'histExpTimeWaitAllocate.svg', format='svg')
502 plt.show()
```
