МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ІВАНОЧКО Вячеслав Романович

Модель енергоефективної системи вводу/виводу на основі SSD і HDD / An energy-efficient Model of I/O system based on SSD and HDD

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи
КНм-21
В. Р. Іваночко
_____

Науковий керівник:
к.т.н., доцент
О.Р. Осолінський
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
Завідувач кафедри
_____ М.П. Комар

ТЕРНОПІЛЬ - 2023

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Іваночку Вячеславу Романовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Модель енергоефективної системи вводу/виводу на основі SSD і HDD / An energy-efficient Model of I/O system based on SSD and HDD
керівник роботи к.т.н., доцент О.Р. Осолінський
затверджені наказом по університету від 8 грудня 2022 року № 491

2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2023 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити
- визначити принципи роботи жорстких та твердотільних дисків;
- дослідити методи моделювання пристроїв зберігання даних;
- дослідити енергозберігаючі рішення для накопичувачів;
- удосконалити метод моделювання чорної скриньки" для пристроїв зберігання даних;
- побудувати модель
- розробити та реалізувати систему оцінки моделювання чорної скриньки;
- розробити та провести моделювання енергозберігаючої архітектури;
- провести тестування моделей на основі реальних трас;

5. Перелік графічного матеріалу у роботі
- Блок - схема реалізації чорної скриньки;

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 8 грудня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Сучасний стан досліджень в області моделювання та оптимізації роботи накопичувачів	12.2022 р. – 03.2023 р.	
2	Моделювання чорної скриньки поведінки пристрою зберігання	03.2023 р. – 05.2023 р.	
3	Реалізація та тестування методу “чорної скриньки”, для енергоощадної архітектури для накопичувачів	05.2023 р. – 11.2023 р.	
4	Повне завершення та представлення кваліфікаційної роботи на кафедру	01.12.2023 р.	

Студент _____ В. Р. ІВАНОЧКО
підпис

Керівник роботи _____ к.т.н., доцент О.Р. Осолінський
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Модель енергоефективної системи вводу/виводу на основі SSD і HDD «Магістр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 103 сторінки і містить 29 ілюстрацій, 4 додатки та 97 використаних джерел.

Метою даної кваліфікаційної роботи є розробка моделі енергоефективної системи вводу/виводу на основі SSD і HDD.

Методи досліджень: методи математич ного і графічного програмування; системного аналізу,системного програмування, методи комп’ютерного моделювання.

Результати дослідження: розробці моделі енергоефективної системи вводу/виводу на основі SSD і HDD.

Результати роботи можуть успішно застосовуватися при створені енергоефективних систем а також при дослідженні та симуляції поведінки накопичувачів при різних режимах роботи.

Ключові слова: SSD і HDD, енергоефективність, модель, система вводу/виводу.

RESUME

The qualification work on the topic " An energy-efficient Model of I/O system based on SSD and HDD "Master's" from the specialty 122 "Computer Science" of the educational program "Computer Science" is written in the volume of 103 pages and contains 29 illustrations, 4 appendices and 97 references.

The purpose of this qualification work is to develop a model of an energy-efficient I/O system based on SSD and HDD.

Research methods: methods of mathematical and graphical programming; system analysis, system programming, computer modeling methods.

Research results: development of a model of an energy-efficient I/O system based on SSD and HDD.

The results of the work can be successfully used in the creation of energy-efficient systems as well as in the study and simulation of the behavior of drives in various operating modes.

Keywords: SSD and HDD, energy efficiency, model, I/O system.

ЗМІСТ

ВСТУП

Протягом багатьох років покращення продуктивності підсистеми вводу/виводу та інших підсистем комп'ютера відбувалося різними темпами, причому темпи покращення продуктивності підсистеми вводу/виводу були меншими за темпи покращення швидкодії підсистеми вводу/виводу, що ставило загальну продуктивність системи в залежність від швидкості підсистеми вводу/виводу.

Одним з головних чинників такого дисбалансу є природа накопичувачів, яка дозволила досягти значних успіхів у підвищенні щільності дисків, але не настільки значних у підвищенні їхньої продуктивності.

Одним з найпоширеніших методів оцінки продуктивності підсистеми вводу/виводу комп'ютера є використання детальних імітаційних моделей, що включають специфічні особливості пристроїв зберігання даних, такі, як геометрія диска, розбиття на зони, кешування, буфери попереднього зчитування та переупорядкування запитів. Однак, як тільки з'являється нова технологічна інновація, ці моделі потрібно переробляти, щоб включити нові характеристики, що ускладнює актуалізацію загальних моделей.

Альтернативним підходом є моделювання пристрою зберігання даних, як імовірнісної моделі "чорної скриньки", де сам пристрій зберігання даних, його інтерфейс та механізми взаємозв'язку моделюються як єдиний процес, визначаючи час обслуговування, як випадкову величину з невідомим розподілом. Такий підхід дозволяє генерувати час роботи диска з меншими обчислювальними витратами за допомогою генератора випадкових подій

Крім того надійність системи зберігання даних має важливе значення, оскільки збої в роботі компонентів системи зберігання можуть призвести до пошкодження або навіть повної втрати даних. Тому, окрім багаторівневого резервування, у дата-центрах поширеною практикою є своєчасна заміна пристроїв зберігання даних.

Виходячи з даних аналізу моделювання робочих навантажень та вимірювання часу обслуговування є актуальною задачею.

Метою роботи є розробка моделі енергоефективної системи вводу/виводу на основі SSD і HDD

Досягнення цієї мети зумовило потребу теоретичних розробок, визначення та послідовного вирішення таких завдань:

1. визначити принципи роботи жорстких та твердотільних дисків;
2. дослідити методи моделювання пристроїв зберігання даних;
3. дослідити енергозберігаючі рішення для накопичувачів;
4. удосконалити метод моделювання чорної скриньки" для пристроїв зберігання даних;
5. побудувати модель
6. розробити та реалізувати систему оцінки моделювання чорної скриньки;
7. розробити та провести моделювання енергозберігаючої архітектури;
8. провести тестування моделей на основі реальних трас;

Об’єктом дослідження є енергоефективні моделі і системи вводу/виводу даних на базі HDD і SSD.

Предметом дослідження є методи побудови моделей, симуляторів для відстеження стану роботи накопичувачів в різних умовах.

Методи дослідження. У роботі використані методи математичного, графічного програмування, системного програмування; системного аналізу; моделювання.

Наукова новизна отриманих результатів полягає у розробці моделі енергоефективної системи вводу/виводу на основі SSD і HDD.

Практичне значення одержаних результатів полягає у створені моделі «чорної скриньки» та енергозберігаючої архітектури для системи вводу/виводу на базі HDD і SSD накопичувачів.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися на

V Міжнародній студентській науковій конференції, «Цифровізація науки та сучасні тренди її розвитку» 24 листопада, 2023 року та на IV Міжнародній студентській науковій конференції, «Тренди та перспективи розвитку мультидисциплінарних досліджень», 1 грудня, 2023 року.

Робота складається із вступу, трьох розділів, висновків, списку використаних джерел. Повний обсяг роботи становить 103 ст. комп'ютерного тексту, який включає 37 рисунків. Список використаних джерел із 97 найменувань викладено на 6 сторінках.

1 СУЧАСНИЙ СТАН ДОСЛІДЖЕНЬ В ОБЛАСТІ МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЇ РОБОТИ НАКОПИЧУВАЧІВ

1.1 Основи жорстких та твердотільних дисків

1.1.1 Жорсткі диски

Протягом багатьох років підсистеми вводу/виводу вважалася вузьким місцем, оскільки їхня продуктивність покращувалася не так швидко, як в інших компонентах архітектури комп’ютерних систем. Крім того, в останні кілька років компоненти підсистем вводу/виводу були визнані одними з найбільш енергоємних частин.

В обох випадках причиною стало виготовлення та конструктивні особливості жорстких дисків, які лежать в основі підсистеми вводу/виводу.

Незважаючи на сучасний стан техніки, жорсткі диски (HDD) (рисунок 1.1) протягом багатьох років були і є найпоширенішими пристроями для вторинного зберігання даних у звичайних комп'ютерах. Сьогодні вони все ще залишаються такими завдяки своїй високій ємності, низькій ціні та зменшенню середнього часу доступу до даних.

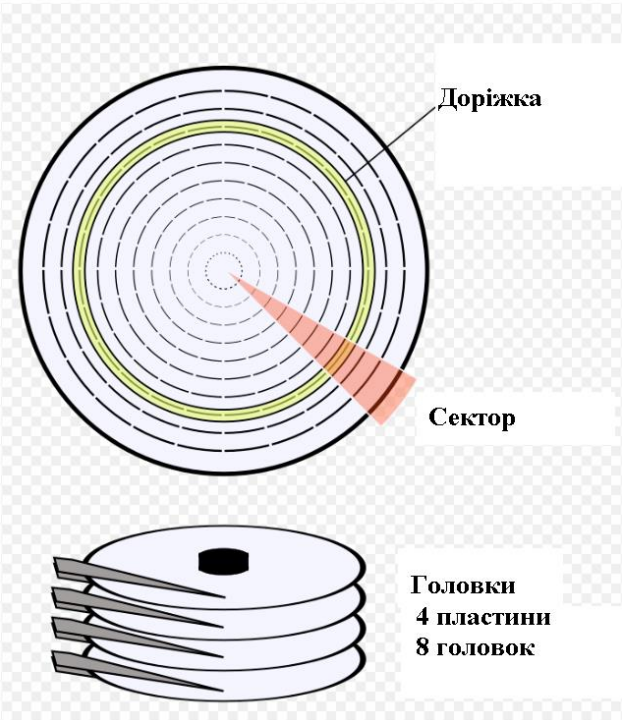


Рисунок 1.1 – Структура HDD

Звичайні дискові накопичувачі мають шпиндель, який утримує диски, на яких зберігаються дані. Головки зчитування і запису зчитують/записують дані на пластини. Рукоятка переміщує головки, дозволяючи їм отримати доступ до всієї пластини.

Зазвичай пластина має дві поверхні, на яких можуть зберігатися дані, і, відповідно, дві головки на пластину. Кожна поверхня організована у вигляді концентричних смуг (так званих доріжок), які мають сектори. Сектор - це найменша адресна одиниця жорсткого диска. Його розмір зазвичай становить 512 байт.

Сукупність паралельних доріжок на кожній поверхні пластини називається циліндром. Типовий диск має стільки циліндрів, скільки доріжок на поверхні пластини. На рисунку 1.1 показано найважливіші частини дискового накопичувача

Час доступу до диска пов'язаний з його конструкцією, а саме механічних частин. Також час доступу можна розділити на 3 частини: Час пошуку, час затримки обертання та час передачі даних.

Час пошуку - це час, за який головка читання/запису досягає доріжки, що містить дані, до яких потрібно отримати доступ. Після досягнення потрібної доріжки потрібен час на затримку обертання - це затримка, яка необхідна для того, щоб головка опинилася під потрібним сектором. Час передачі даних - це час, необхідний головці для зчитування запитуваних даних. Чим більше даних потрібно зчитати, тим більше часу затрачається.

Доступ до пластин зазвичай займає кілька мілісекунд. Щоб розподілити цей ефект, дискові накопичувачі мають внутрішню кеш-пам'ять, яка обслуговує запити набагато швидше, ніж безпосередньо з пластин. Коли на диск надходить запит на читання, спочатку перевіряється кеш диска на наявність запитуваних даних. Якщо запитувані дані є в кеші, вони обслуговуються з нього, а звернення до пластин уникається. І навпаки, якщо запитуваних даних немає в кеші диска, запитувані дані обслуговуються з дискових пластин, а потім зберігаються в кеші. Одним із застосувань кешу читання диска є попередня вибірка. Коли вона

активована, кеш завантажує дані, до яких ще не було доступу, але є ймовірність, що до них буде звернення. Коли на диск надходить запит на запис, а кешування увімкнено, дані, що записуються, безпосередньо записуються в буфер диска, що підвищує продуктивність. Ці дані зазвичай пізніше записуються на пластини SAS і Fibre Channel. Інші інтерфейси, такі як IEEE 1394 і USB, використовують мостову схему для зв'язку з дисковим накопичувачем.

До дискових накопичувачів можна отримати доступ через кілька типів інтерфейсів, таких як IDE, SATA, SCSI,

З метою економії електроенергії сучасні диски мають кілька станів живлення: активний, бездіяльності і режим очікування. Диск знаходиться в активному стані, коли він читає або записує дані. Коли диск нічого не робить, але продовжує обертатися, він перебуває у стані очікування. Коли диск нічого не робить і його пластини також не обертаються, він перебуває у стані бездіяльності. Найбільше енергоспоживання відбувається в активному стані.

Стан бездіяльності споживає менше енергії, ніж активний стан, а стан очікування споживає набагато менше енергії, ніж попередні два стани. Диск переходить у стан очікування, коли він перебуває в режимі очікування протягом певного періоду часу і передбачається, що зміна стану є доцільною.

Коли диск, який перебуває в стані очікування, повинен обслужити запит на введення/виведення, він повинен розкрутитися до активного стану, а потім обслужити запит на введення/виведення. Розкручування диска займає кілька секунд і вимагає витрат енергії.

Кожного разу, коли диск обертається вниз і вгору, головки і двигун шпинделя зношуються. Тому виробники вказують максимальну кількість циклів запуску/зупинки, яку дисковий накопичувач може витримати без помилок. Для накопичувачів для настільних комп'ютерів цей показник становить близько 50 000 циклів запуску/зупинки, а для ноутбуків - близько 300 000.

Вартість SATA-накопичувача ємністю 3.5 ТБ становить близько 1700 грн, тобто близько 0,48 грн. за ГБ [1].

1.1.2 Твердотільні диски

Твердотільні диски (SSD) - це пристрої, які використовують флеш-пам'ять для вторинного зберігання даних у звичайних комп'ютерах. Вони легкі, ударостійкі, безшумні та споживають менше енергії, оскільки не мають механічних частин.

Основними частинами SSD є інтерфейс, флеш-пам'ять, контролер і кеш-пам'ять. На рисунку 1.2 показано огляд найважливіших частин твердотільного накопичувача.

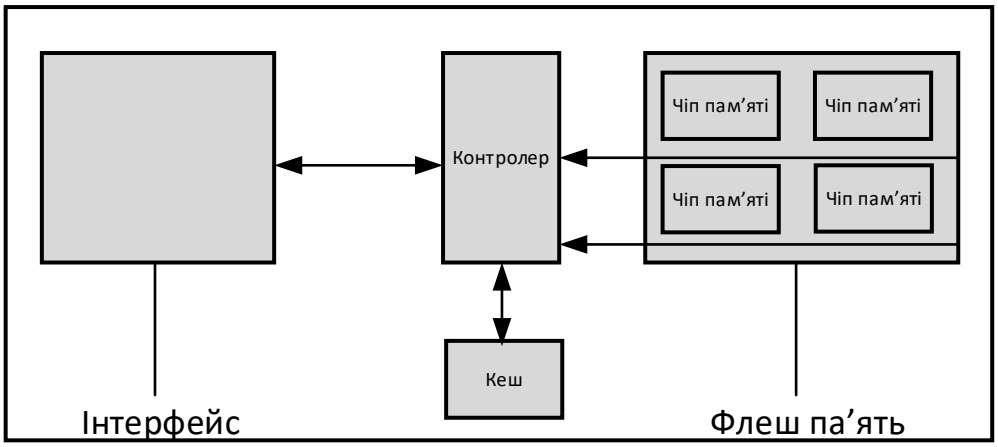


Рисунок 1.2 – Твердотільний еакопичувач

Інтерфейс дозволяє твердотільному накопичувачу підключатися до комп'ютера і поводитися так само, як і жорсткий диск. Інтерфейси можуть бути такими, як: SATA, SAS, PCI Express, Fibre Channel, USB, IDE і SCSI.

Флеш-пам'ять (рисунок 1.3) зазвичай має тип NAND через свою низьку вартість.

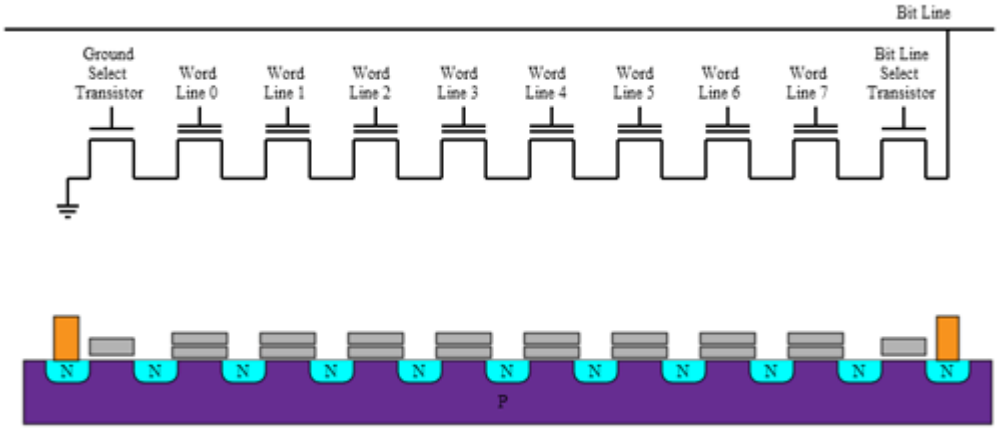


Рисунок 1.3 – Структура флеш-пам'яті

Вона складається з декількох флеш-пакетів. Кожен пакет має одну або кілька мікросхем пам'яті. Кожна мікросхема складається з декількох блоків, а кожен блок містить кілька сторінок.

Зазвичай розмір блоку становить 16-256 КБ, а розмір сторінки - 0,5-4 КБ. Дані зчитуються/записуються з/на сторінки. Читання сторінки займає близько 20-25 мкс, а запис сторінки - 200-300 мкс [2]. Перш ніж сторінку можна буде використати повторно, її потрібно стерти. Стирання відбувається на рівні блоків, тому воно займає більше часу, близько 1,5-2 мс.

Контролер має необхідну схему для з'єднання флеш-пам'яті з хост-комп'ютером. Серед іншого, він містить складний модуль, який називається Flash Trans lation Layer (FTL) [3,4]. Однією з його важливих функцій є максимізація паралелізму, який можуть забезпечити мікросхеми флеш-пам'яті. Доступ до декількох мікросхем може здійснюватися по черзі, що підвищує продуктивність SSD, так само, як це організовано в RAID-масиві для дискових накопичувачів. Інша важлива роль FTL полягає у відображенні секторів у флеш-пам'яті. Існує кілька підходів, наприклад, відображення сторінок і блоків. У Nand флеш-пам'яті звичайним підходом є блокове відображення. Також існують гібридні підходи [5, 6].

SSD також мають кілька мікросхем кеш-пам'яті, які схожі на кеш-пам'ять в дискових накопичувачах. Вони допомагають досягти кращої продуктивності. Оскільки твердотільні накопичувачі не мають механічних частин, вони мають

лише два стани живлення: активний і неактивний. SSD знаходиться в активному стані, коли він читає або записує дані. Коли SSD нічого не робить, він перебуває в режимі очікування.

SSD можна записувати обмежену кількість разів. Щоб перезаписати певний блок, його потрібно спочатку стерти, тобто кількість циклів запису-стирання в даних накопичувачах є обмеженою, і коли досягається максимальна кількість стирань, блоки SSD стають недоступними для стирання, але їх все ще можна прочитати.

Пам'ять NAND можна розділити, як показано на рисунку 1.4 [7,8]: Однорівневі комірки (Single Level Cell, SLC, зберігає 1 біт даних), багаторівневі комірки (Multi Level Cell, MLC, зберігає 2 біти даних) і трирівневі комірки (Triple Level Cell, TLC, зберігає 3 біти даних).

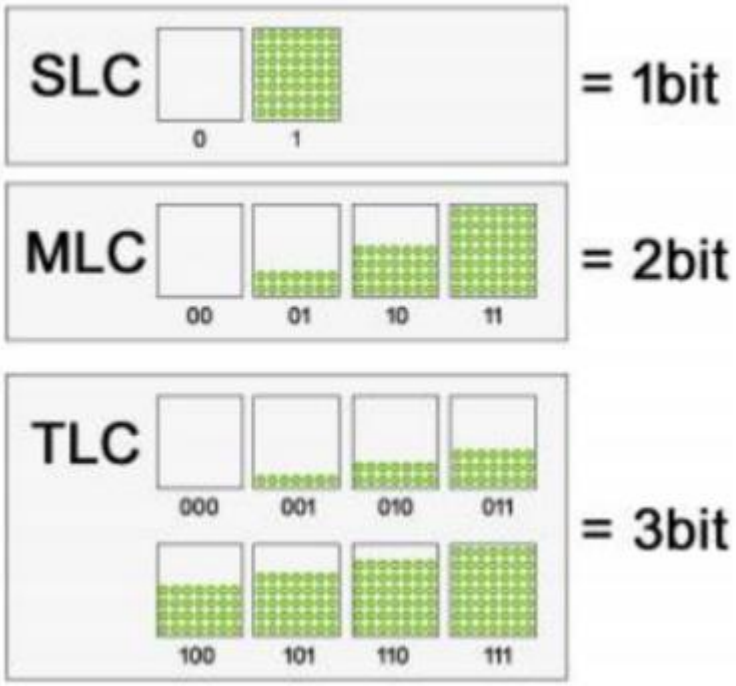


Рисунок 1.4 - Типи флеш пам’яті

Підвищена ймовірність помилок також обмежує кількість разів, яку можна записати певний блок, що становить близько 10 000 разів для флеш-пам'яті MLC, 100 000 для флеш-пам'яті SLC і 300 000 для TLC.

Вартість твердотільного накопичувача на основі SLC-пам'яті ємністю 512 ГБ становить близько 1300 грн, що дорівнює приблизно 0,39 грн. за ГБ [9, 10].

Для MLC – 500 Гб = 1900 грн. що становить 0,26 грн за ГБ і TLC 500 ГБ становить 0,38 грн.

1.2 Моделювання пристроїв зберігання даних

Традиційним підходом до моделювання пристроїв зберігання даних є аналітичні моделі. Вони використовують математичні рівняння для опису поведінки диска. Багато з цих ідей було втілено у симуляторах, які також дуже детально імітували багато характеристик дисків, досягаючи у багатьох випадках рівня емуляції. Але, обидва підходи вимагають глибоких знань про внутрішню будову пристрою, який потрібно сконструювати. Альтернативний підхід полягає у моделюванні поведінки диска, як би він був «чорною скринькою». Диск, разом з його інтерфейсом та механізмами з'єднання, можна розглядати, як об’єкт, деталі якого невідомі.

1.2.1 Аналітичні моделі

Аналітичні моделі використовують математичні рівняння для опису характеристик сховища. На відміну від симуляторів, ці описи є своєрідним резюме того, як працюють окремі частини сховища. Тому вони дуже швидкі. Однак для деяких частин пристроїв зберігання даних отримати рівняння, яке їх описує досить складно.

Алгоритми планування роботи дисків були запропоновані для того, щоб зменшити кількість переміщення дискових маніпуляторів, особливо під час пікових навантажень, обробляючи запити до дисків у такому порядку, щоб їм доводилося чекати якомога менше часу. Класичними найпоширенішими алгоритмами планування є First Come First Served (FCFS), Sortest Seek Time First (SSTF) і SCAN. В роботах [11] були запропоновані та досліджені нові алгоритми планування, які покращують попередні, для включення в інші аналітичні/детальні моделі або реальні пристрої.

Автори в роботі [12, 13] описали модель, що складається з декількох незалежних моделей. Кожна модель містить формули для певної частини дискового накопичувача, такі як черги, кешування та механізми роботи дискового накопичувача. Вхідними даними для глобальної моделі є робоче навантаження, яке трансформується через усі модулі до отримання часу обслуговування. Автори також розробили - метод прогнозування переупорядкування даних на випередження, який враховує вплив на продуктивність послідовних звернень до диска, який є побічним ефектом дослідження в цій роботі.

В роботі [14, 15] запропоновано аналітичну модель, яка аналізує декілька дискових технологій. Наведено формули для моделювання оптичних дисків з концентричними доріжками. Для магнітних дисків виведено формули для затримок пошуку та обертання з урахуванням як змінної, так і постійної швидкості обертання. Основним внеском роботи є моделювання нових введених зонованих дисків.

Опис аналітичної моделі пропускної здатності дискових масивів наведено в роботі [16]. Як і в [17], вона є дуже модульною і може бути декомпонована для опису різних частин дискових масивів. Серед них модулі для дисків, шин і контролерів. Вся модель порівнюється і перевіряється за допомогою вимірювань, зроблених на реальних дискових масивах. Вона також включає в себе кілька сучасних оптимізацій, які часто зустрічаються в сучасних дискових масивах. Модульна конструкція полегшує додавання нових модулів.

Аналітичну модель часу обслуговування в дискових масивах запропоновано в [18]. Коли один запит звертається до масиву дисків, він може бути розбитий на декілька запитів, час обслуговування яких дорівнює максимальному з часів обслуговування підзапитів. Сама модель побудована на простих рівняннях.

Робота, запропонована в [19], описує нову стратегію стрипінгу для дискових масивів і нову політику черги на дисках. Аналітична модель

будувалась на основі розгляду окремих дисків, як незалежних моделей і для кожного з них була окрема модель черги [20].

Інша аналітична модель для дискових масивів запропонована у [21]. Представляє модель продуктивності дискових масивів під час синхронних робочих навантажень вводу/виводу. Модель перевірено на реальному дисковому масиві. Модель також включає багато типових характеристик, притаманних сучасним дисковим масивам, таких як кешування з випередженням читання та зворотним записом, планування дисків, об'єднання запитів та надлишковість у масиві. Також моделюються ефекти послідовності та паралелізму. Недоліком моделі є те, що вона була перевірена лише для робочих навантажень, що містять лише читання або лише запис.

Як було сказано раніше комірки пам'яті в SSD можуть бути записані обмежену кількість разів. Отже, розумне управління внутрішніми комірками цих пристроїв може значно подовжити їхнє життя. Математична модель для задач управління флеш-пам'яттю визначена в [22]. Також проаналізовано та порівняно декілька алгоритмів вирівнювання зносу.

Підсилення запису тим більше, чим більше внутрішніх записів доводиться робити в SSD, щоразу, коли вони записуються. Виходячи з результатів моделювання, можна зробити висновок, що вирівнювання зносу є важливим ключем до продовження терміну служби твердотільних накопичувачів.

В роботі [23] запропоновано дві аналітичні моделі, що описують характеристики твердотільних накопичувачів. Перша - це загальна модель флеш-пам'яті, а друга - модель одиничної флеш-пам'яті. Обидві моделі описують різні характеристики флеш-накопичувачів і є корисними для розробки нових моделей.

Авторами в роботі [24], описано ще одну аналітичну модель для оцінювання продуктивності двох різних типів надшвидкісних накопичувачів у твердотільних накопичувачах. На відміну від інших подібних моделей, запропонованих раніше, вона використовує реальні робочі навантаження. Ця модель є розширенням моделі, запропонованої в [25].

1.2.2 Симулятори

Симулятори моделюють пристрої з високою деталізацією та поведінкою наближеною до реальних накопичувачів, тому розробники симуляторів повинні добре розуміти внутрішню структуру пристроїв зберігання даних.

У деяких випадках симулятори можуть досягати рівня емуляції. Ця характеристика робить їх дуже точними, але якщо потрібно оцінити масштабовані системи зберігання даних, то обчислювальна потужність зростає зі збільшенням кількості пристроїв, які включені в змодельовану систему. В результаті використання детальних симуляторів може негативно вплинути на час виконання при запуску довгих симуляцій.

Програма DiskSim [26] – це симулятор підсистеми зберігання даних, що включає модулі для моделювання кеш-пам'яті, шин, контролерів, драйверів пристроїв та дискових накопичувачів. Накопичувач, який у цьому симуляторі було деталізовано, був перевірений на багатьох дисках і навіть покращив інші попередні симулятори [27]. Решта компонентів (шини, адаптери, контролери та драйвери) не були перевірені [28]. DiskSim використовує багато параметрів, які отримуються з накопичувачів за допомогою напівавтоматичних алгоритмів [29], або за допомогою інструменту характеризації дисків DIXtrac [30]. Слід зазначити, що DIXtract отримує понад 100 критичних параметрів продуктивності. Однак DIXtract може отримувати параметри лише з SCSI дисків.

Симулятор Pantheon [31] – це програма-симулятор підсистеми вводу/виводу, який спочатку використовувався для моделювання паралельних дискових масивів. Дана програма імітує такі компоненти, як диски, шини та контролери масивів. В ній перевірені лише дискові модулі [28]. Також її було розширено для моделювання як однопроцесорних, так і паралельних систем.

Програма RAIDFrame [32] також було створено для оцінки масивів дискових архітектур. Це програмний RAID-контролер, який також можна використовувати як окремий симулятор дискретних подій для дискових масивів [33].

Програмний пакет FlashSim [34] призначений для оцінки систем зберігання даних, що використовують SSD. Його можна використовувати з різними схемами FTL і він легко інтегрується з DiskSim. Ще один симулятор від Microsoft [35] також симулює SSD і інтегрований з Disksim. Він реалізує методи чергування, доступні в звичайних SSD-пристроях, але дозволяє використовувати лише одну схему FTL. У [36] розроблено ще один симулятор SSD, але він не дозволяє тестувати різні схеми FTL і не може бути інтегрований з Disksim.

1.2.3 Моделі "чорних скриньок

Моделі "чорної скриньки" припускають, що про пристрій зберігання даних майже нічого не відомо. Пристрої та всі їхні характеристики розглядаються, як чорні скриньки, де внутрішні деталі невідомі. Серед інших переваг можна назвати те, що вони, як правило, швидкі, оскільки не дуже деталізовані. Їх легко розробляти і оновлювати, оскільки не потрібно знати внутрішню будову пристроїв і нові технології; і вони можуть бути також достатньо точними.

Найпростішими моделями "чорних скриньок" є моделі на основі таблиць [37]. Вони просто пов'язують у таблицях пам'яті інформацію про вхідні дані від робочих навантажень з вихідними даними про реальний час обслуговування. Вхідними даними від робочих навантажень можуть бути розміри запитів або типи запитів. Вихідні дані - це отриманий час обслуговування при відтворенні трас на реальних пристроях. Чим більші таблиці, тим точніші моделі. Відсутні дані в таблицях просто інтерполюються з найбільш схожих характеристик в таблицях. Точність можна підвищити, додаючи нову інформацію до таблиць. Однак це рішення не є масштабованим, оскільки, чим більше інформації додається до таблиць, тим повільніший пошук в таблицях і тим повільніший прогноз часу обслуговування.

В роботі [38] пропонується вдосконалення роботи [36]. В дослідженні автори не використовують таблиці, а застосовують CART (Classification And Regression Tree) [39] моделювання для пристроїв зберігання даних. У CART-моделях цільові/вихідні дані організовані в дерева, доступ до «листя» яких

здійснюється за допомогою інформації з вхідних даних, таких, як розміри або типи запитів, або інші характеристики робочих навантажень. Вони пропонують два підходи:

- перший прогнозує час відповіді на кожен запит;
- другий прогнозує агрегований час відповіді.

Підхід для кожного запиту прогнозує час відгуку, беручи до уваги лише характеристики цього запиту. Агрегований підхід прогнозує, беручи до уваги характеристики всього робочого навантаження. Предиктор для кожного запиту також більш вимогливий до обчислень, ніж агрегований предиктор. З іншого боку, предиктор на запит може досягти кращої точності. Оцінки моделі не використовують реальні диски. Замість цього використовуються екземпляри симулятора DiskSim.

Автори роботи [40] оцінюють точність моделі чорного ящика для жорстких дисків на основі дерев регресії. Вони використовують алгоритм з відкритим вихідним кодом під назвою GUIDE [41]. Цей алгоритм використовує аналіз залишків за критерієм Хі-квадрат і усуває зсув у виборі змінних. Оцінки включають два типи середовищ:

- середовище з одним робочим навантаженням;
- середовище з декількома робочими навантаженнями.

Середовище з одним робочим навантаженням імітує один потік робочого навантаження, тоді як середовище з декількома робочими навантаженнями імітує сценарій з декількома потоками. Автори дійшли висновку, що сценарій з декількома робочими навантаженнями важче передбачити.

У роботі [42] продуктивність пристрою зберігання даних моделюється шляхом підстроювання характеристик продуктивності, використання ресурсів і робочого навантаження іншого пристрою, на якому базується перший пристрій. Подібні або не подібні пристрої можуть бути пов'язані певним чином: Вони можуть мати схожі апаратні характеристики або ні, вони можуть вести себе подібним чином при однакових робочих навантаженнях. Основна мета пропонованого вище методу - дізнатися про зв'язок між пристроями, щоб

побудувати похідну модель на основі вже змодельованого пристрою. Метод починається з моделі абсолютного чорного ящика, яка будується за допомогою моделювання CART [43]. Недоліком цієї моделі є те, що якщо абсолютна модель не є точною, помилки можуть передаватися на похідну модель, роблячи її менш точною.

Автори робіт [44] використовують дерева регресії, як і в [45], для моделювання "чорної скриньки" SSD-пристроїв. Вони запропонували два підходи:

- базову модель;
- розширену модель.

Обидві моделі отримують на вхід характеристики робочого навантаження і отримують прогнози продуктивності. У базовій моделі робоче навантаження характеризується відсотком записів/зчитувань, розмірами запитів, кількістю попередніх запитів, що очікують на виконання, та відсотком випадкових запитів.

У розширеній моделі характеристики навантаження ті ж самі, але розділені з урахуванням того, чи є це читання або запис, через асиметричну продуктивність твердотільних накопичувачів. Крім того, вивчаються дві інші моделі, відмінні від випадкової, такі як послідовна та стрічкова. Однак оцінки не використовують реальні траси. Вони лише використовують синтетичні траси з раніше згаданими, попередньо визначеними шаблонами.

В роботі [46] запропоновано модель енергоспоживання "чорного ящика" для дискових масивів. Тут дисковий масив разом з контролером і декількома дисками розглядається, як чорний ящик, споживання енергії якого можна виміряти.

1.3 Енергозберігаючі рішення для накопичувачів

Рішення з енергозбереження, спрямовані на економію електроенергії в комп'ютерних системах. Накопичувачі вважаються одними з енергоємних елементів комп'ютерної системи, особливо в контексті підсистеми вводу/виводу.

Традиційні дослідження енергозбереження пропонують методи для економії енергії в накопичувачах для ноутбуків/персональних комп'ютерів. Останнім часом енергозбереження в корпоративних системах зберігання даних також викликало багато досліджень.

1.3.1 Керування живленням, орієнтоване на ноутбуки/стаціонарні комп'ютери

Ноутбуки, планшети та термінали [47] можуть працювати в автономному режимі лише кілька годин, перш ніж закінчиться заряд акумулятора. Тільки декілька компонентів таких систем дійсно впливають на енергоспоживання, а саме:

- дисплей був визначений, як найбільш енергоємний елемент,
- накопичувач – другий по енергоспоживанню елемент [48].

Більшість технологій використовують певні періоди бездіяльності, щоб зупинити компоненти і заощадити енергію, коли вони не використовуються. Ця технологія є найбільш поширеною в контексті накопичувачів. Отже, мета тут полягає в тому, щоб заощадити енергію акумулятора, зупинивши жорсткий диск або перевести в режим очікування твердотільний накопичувач, і таким чином збільшити час роботи комп'ютера.

1.3.2 Політика вимкнення

Як було сказано раніше, диски можуть заощаджувати енергію, якщо їх зупиняти. Зупинка диска означає переведення його у режим зниженого енергоспоживання. Дискові накопичувачі можуть перебувати у трьох режимах: активному (пластини обертаються, а головки обслуговують запити), холостому (пластини обертаються, але головки не обслуговують запити) і режимі очікування (пластини не обертаються, а головки знаходяться у режимі очікування, заощаджуючи енергію). Коли диск перебуває в режимі очікування протягом певного періоду часу, він переходить у режим очікування. Як тільки на диск надходить запит, він розкручується до активного режиму, щоб

обслужити цей запит. Оскільки процеси згортання і розгортання споживають час і енергію, диск повинен перебувати у стані очікування принаймні мінімальний проміжок часу, щоб ці зміни були виправдані і щоб диск заощаджував енергію. Політика вимкнення визначає, коли вимикати диск, щоб заощадити енергію.

Автори роботи [49] надають аналіз компромісів і переваг пригальмовування диска, як метод зниження енергоспоживання. Перш за все, з'ясовується найбільш підходяща затримка пригальмовування (тривалість часу, протягом якого диск очікує на подальшу активність, перш ніж перейти в стан очікування). Продемонстровано, що затримка розкручування в 2 секунди дає найбільшу економію енергії. По-друге, проаналізовано компроміс між енергоспоживанням і затримкою користувача для попередньої 2-секундної затримки розгортання. Результати показують, що користувачеві доведеться чекати 15-30 секунд на годину, що відповідає 2-секундному розгортанню диска приблизно 8-15 разів на годину. Далі проаналізовано використання дискового кешу з точки зору енергозбереження. Продемонстровано, що використання кешу розміром в один мегабайт є достатнім для досягнення більшості енергетичних переваг дискового кешування, а кеш більшого розміру не дасть ніякої додаткової економії. Більше того, затримка запису пошкоджених блоків з кешу на диск на 30 секунд дасть додаткову економію енергії, а більша за цю величину затримка не дасть додаткової економії. Іншим проаналізованим питанням є той факт, що ця технологія призводить до збільшення часу обертання диска і збільшує знос інтерфейсу диск-головка. Ця проблема вирішується в наступному поколінні дискових накопичувачів, які дозволяють більш тривале безперервне використання. Нарешті, оскільки до імен та атрибутів файлової системи часто звертаються, також пропонується використовувати кеш імен та атрибутів. Його використання дозволяє досягти більшої економії енергії.

В [50] запропоновано декілька адаптивних політик призупинення. Користувач повинен визначити рівень прийнятності, який є метрикою, що показує кількість небажаних розгортань, які користувач може допустити. Якщо користувач визначає високий рівень прийнятності, то буде здійснено багато

зупинок за рахунок компромісу між продуктивністю та зменшенням енергоспоживання. І навпаки, якщо користувач встановлює низький рівень прийнятності, буде менше небажаних розкручувань, економія енергії зменшиться, а продуктивність буде кращою.

Коригування порогу відбувається під час адаптивної політики або коли відбувається неприйнятне розкручування, або коли інша інформація вказує на необхідність змінити поріг. Адаптивне згортання порівнюється з політикою з фіксованим порогом. Результати показують, що:

- іноді політика адаптивного згортання усуває значну частину небажаних згортань з невеликим збільшенням енергоспоживання;

- в інших випадках небажані розгортання залишаються такими ж, як і при зміні політики фіксованого порогу;

- іноді результати є гіршими, ніж при використанні політики фіксованого порогу.

Важливо відзначити, що у деяких випадках ефективність адаптивних політик згортання залежить від використовуваної траси та диска.

У роботі [51] проблема, підключення накопичувача у ноутбуках, вирішується за допомогою алгоритму машинного навчання - алгоритму спільного використання (share algorithm). Він отримує на вході набір експертів, які роблять прогнози. Алгоритм об'єднує прогнози експертів і присвоює кожному експерту одну вагу. Ця вага відображає якість прогнозів кожного експерта. Ваги експертів постійно оновлюються. Результати показують, що алгоритм частки кращий за інші практичні алгоритми, які зазвичай застосовуються, такі як

- алгоритм з фіксованою затримкою віджиму, який обирає одну фіксовану затримку віджиму як значення і починає віджимати диск після того, як він простоював протягом цього періоду,

- фіксована затримка віджиму, що дорівнює вартості віджиму дискового алгоритму, або

- рандомізовані алгоритми, які вибирають значення затримки віджиму з деякого розподілу.

Реалізація алгоритму є досить ефективною і може бути реалізована на контролері накопичувача.

1.3.3 Енергоощадні методи попередньої обробки та кешування

Правильний вибір часу вимкнення накопичувача допомагає заощадити енергію. Коли диск перебуває у стані бездіяльності протягом певного періоду часу (встановленого політикою вимкнення), він переходить у стан очікування. Як тільки на диск надходить запит, він повинен активуватись, щоб обслужити його, незалежно від політики вимкнення. Методи попередньої вибірки та кешування, які зменшують кількість звернень до диска, стають дуже корисною підтримкою для політики згортання, щоб заощадити ще більше енергії. Коли на диск надходить запит, а він знаходиться в стані очікування, якщо запит знаходиться в кеші, то спін-апу можна уникнути, а час простою диска може бути довшим, що дозволяє заощадити більше енергії. Основна мета цих підходів - максимізувати час простою диска за допомогою кешування, щоб якомога довше тримати диск у режимі очікування.

Кешування з попередньою вибіркою зазвичай використовується для приховування дискових затримок, але всерівно воно не зменшує енергоспоживання дисків. Але автори роботи [52] запропонували алгоритм попередньої вибірки на основі епох в механізми управління пам'яттю операційної системи з метою енергозбереження.

Кожна епоха складається з двох фаз: фази простою та активної фази. Під час фази очікування накопичувач працює в режимі зниженого енергоспоживання, доки не буде ініційовано новий цикл попередньої вибірки, не відбудеться пропуск або система не вичерпає ресурси пам'яті.

У цій фазі очікування доступу до кожного активного файлу контролюється операційною системою за допомогою «демонів» (Linux), які намагаються передбачити наступну частоту пропусків для кожного файлу.

Під час активної фази виконується попередня вибірка, використовуючи інформацію, яку контролюють демони, і прогнозується тривалість майбутньої фази простою. Крім того, координується доступ паралельно запущених програм, щоб вони отримували дані на диску приблизно в один і той самий час.

Основна мета використання алгоритму - максимізувати час простою диска за рахунок збільшення патернів вводу/виводу. Результати експериментів показують, що чим більший обсяг пам'яті використовується системою, тим більша економія енергії. Економія енергії досягає 60-80%.

В [53] описано дві політики кешування для економії енергії. Обидві політики намагаються максимізувати час простою накопичувача, змінюючи схему доступу до HDD/SSD на кластерну.

Перша, HC-Burst, є політикою заміни, яка обирає для заміни більш кластеризовані блоки та блоки, до яких малоймовірно отримати доступ, визначаючи їхню завантаженість у попередні періоди часу. Кластерні звернення до блоків не порушують інтервалів простою, тому що всі блоки отримують доступ через короткі проміжки часу.

Друга, PC-Burst, - це політика заміни, яка намагається передбачити звернення до диска в майбутньому. Використовуючи цю інформацію, деякі звернення до диска можуть бути витіснені, щоб не порушувати тривалий час простою. Реальна реалізація дозволяє заощадити до 35% енергії, а втрата продуктивності є мінімальною.

Режим ноутбука [54] - це параметр, який постачається з ядром Linux і дозволяє змінювати спосіб розподілу запитів на дисковий ввід/вивід у часі. Зазвичай, Linux розподіляє дисковий ввід/вивід плавно, не дозволяючи диску обертатися протягом тривалих періодів часу.

У ноутбуці, щоб подовжити час роботи від батареї, було б корисно сконцентрувати запити на короткі проміжки часу, з довгими інтервалами бездіяльності між ними. Таким чином, коли ввімкнено режим "заощадження енергії", можна подовжити періоди бездіяльності до десяти хвилин. Під час періодів активності може виконуватися читання на випередження до 4 МБ, що

дозволяє уникнути деяких спін-апів. Крім того, запис може залишатися в пам'яті до тих пір, поки не закінчаться періоди бездіяльності або поки не закінчиться пам'ять.

В роботі [55] використовується енергоефективна система віртуальної пам'яті з флеш-пам'яттю, як основне сховище. Запропоновано дві різні методики:

– за допомогою файлу підкачки

– за допомогою керування кешем пам'яті.

Метод файлу підкачки полягає в тому, що сторінка віртуальної пам'яті розбивається на кілька флеш-сторінок, і при збої сторінки до флеш-пам'яті записуються лише "брудні" підсторінки. При управлінні кешем пам'яті в SRAM кешуються тільки записи, і тут є два підходи:

– перший зберігає в кеші сторінки, які найчастіше і найостанніше використовувалися (політика TF).

– другий враховує проблему накладних витрат, намагаючись розподілити дані, до яких зверталися близько в часі, в один і той самий флеш-блок, щоб «блок-жертва» з невеликою кількістю сторінок міг бути знайдений, коли спрацьовує (TF-Locality).

Метод підкачки дозволяє заощадити до 20% енергії для флеш-пам'яті в середньому, і до 24% для багатопрограмних робочих навантажень. Енергозбереження при використанні політики TF становить до 19,5%. Разом ці дві технології дають економію енергії до 35,6% в середньому для флеш-пам'яті.

1.3.4 Зовнішнє кешування

Зовнішнє кешування відрізняється від кешування у пам'яті тим, що при зовнішньому кешуванні використовується енергонезалежний накопичувач [56]. Тому до нього застосовуються політики кешування або попереднього кешування для економії енергії на диску. Використання енергонезалежних пристроїв зберігання даних, таких як NAND флеш-пам'ять, як кеш для дискових накопичувачів, має кілька переваг над буферною пам'яттю. По-перше, вони мають високу щільність і тому можуть зберігати багато даних. По-друге, вони

енергонезалежні, а це означає, що при відключенні електроенергії дані не будуть втрачені. По-третє, вони мають низьке енергоспоживання. Однак, NAND флеш-пристрої все ще не настільки швидкі, як кеш-пам'ять.

Модуль EXCES [57] - це динамічно завантажуваний модуль ядра Linux, який працює між рівнями файлової системи та планувальника вводу/виводу в стеку пам'яті. Він складається з п'яти компонентів: трекера доступу до сторінок, індикатора, тригера реконфігурації, планувальника реконфігурації та реконфігуратора. Основне завдання модуля – це перенаправлення якомога більше запитів на флеш-накопичувач і утримання накопичувача у сплячому режимі якомога довше. Таким чином, флеш-накопичувач діє як своєрідний кеш для диска.

В [58] запропоновано дисковий кеш на основі флеш-пам'яті. У цьому підході флеш використовується, як спосіб подолання розриву в затримках між жорстким диском і оперативною пам'яттю, заощаджуючи при цьому енергію в середовищі серверного домену. Дисковий кеш на основі Flash розділений на дві області, одна для читання, а інша для запису. Крім того, пропонується програмований контролер флеш-пам'яті, щоб підвищити надійність флеш-комірок і продовжити термін служби пам'яті.

Як в [59], так і в [60] флеш-накопичувач розглядається, як кеш для одного дискового накопичувача в середовищі ноутбука/ПК. В [61] запропоновано дуже простий алгоритм послідовної попередньої вибірки. В [59] запропоновано складніший алгоритм попередньої вибірки, який використовує інформацію з ядра. Тут ядро надає підказки про шаблони доступу до файлів. Також в обох випадках застосовуються методи кешування.

1.3.5 Міграція даних між дисками

Деякі технології створюють достатню бездіяльність на деяких дисках шляхом міграції даних з них на інші диски.

Технологія Write-off Loading [62] - це технологія, яка дозволяє тимчасово перенаправляти запити на запис на накопичувачах, що вивільняються, на

постійне сховище в іншому місці центру обробки даних (ЦОД). Це збільшує економію енергії до 45%-60%.

Обладнання, що використовується для розміщення вивантажених блоків, є кінцем кожного існуючого тому. Кожен том має власного програмного менеджера, який вирішує, коли обертати диски вгору або вниз і куди вивантажувати записи. Було проаналізовано траси на рівні блоків з корпоративного дата-центру, і висновком з цього було те, що існує достатньо часу простою для того, щоб деякі томи можна було обертати вниз, не видаляючи записи з трас. Таким чином, обидва випадки показують, що є час простою, коли томи можуть обертатися і заощаджувати енергію.

У [63] представлено аналіз компромісів і переваг, щоб зрозуміти, чи варто замінювати диски на SSD в корпоративному середовищі. Розглянуто кілька можливостей: збереження поточної конфігурації, що базується лише на дискових накопичувачах, поєднання SSD з дисковими накопичувачами та повна заміна дискових накопичувачів на SSD.

MAID (Massive arrays of idle disks) [64] - це великі масиви зберігання даних, призначені для зменшення витрат на електроенергію при збереженні продуктивності в суперкомп'ютерних середовищах зберігання архівів. У таких середовищах записується велика кількість даних, до яких більше ніколи не звертаються, тому вони не потребують високої продуктивності або підвищеної надійності. Проаналізовано кілька конфігурацій попередньої системи, і результати показують, що зберігається та сама продуктивність, але при цьому досягається економія енергії на 85%.

1.4 Постановка задачі

Враховуючи розвиток та збільшення продуктивності підсистеми вводу/виводу та інших складових компютерних систем можна зробити висновок, що проглядається досить суттєва нерівність між накопичувачами та наприклад шинами даних або ЦП, причому останні програють цю гонку удосконалень. Це

призвело до того що системи на базі SSD і HDD почали суттєво впливати на усі елементи КС.

Одним з головних чинників такого дисбалансу є природа накопичувачів, яка дозволила досягти значних успіхів у підвищенні щільності дисків, але не настільки значних у підвищенні їхньої продуктивності.

Є безліч методів оцінки систем вводу виводу, але самі популярні це - імітаційні моделі, моделі "чорної скриньки", детальних аналітичних моделей. Крім того накладаються енергетичні витрати, наприклад накопичувачі відповідають у деяких випадках за половиеу енергоспоживання системи.

Тому є потреба в розробці методу створення моделі чорної скриньки з урахуванням енергії споживання для накопиувачів які не мають повної інформації про їхню структуру та параметри.

Метою дипломної роботи є роботи є розробка моделі енергоефективної системи вводу/виводу на основі SSD і HDD

Досягнення цієї мети зумовило потребу теоретичних розробок, визначення та послідовного вирішення таких завдань:

1. визначити принципи роботи жорстких та твердотільних дисків;
2. дослідити методи моделювання пристроїв зберігання даних;
3. дослідити енергозберігаючі рішення для накопичувачів;
4. удосконалити метод моделювання чорної скриньки" для пристроїв зберігання даних;
5. побудувати модель
6. розробити та реалізувати систему оцінки моделювання чорної скриньки;
7. розробити та провести моделювання енергозберігаючої архітектури;
8. Провести тестування моделей на основі реальних трас;

Завдання 1-5 розглянуто в параграфах 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, відповідно, а виконання завдань 6-8 представлені у параграфах 3.1-3.3.

Висновки до розділу 1:

В першому розділі даної роботи було досліджено принципи роботи Жорстких та твердотільних накопичувачів, як самих популярних носіїв інформації сьогодні. Також у даному розділі розглянуто засоби та методи моделювання накопичувачів по поведінці по типу чорної скриньки. Окрім того було проаналізовано можливість використання аналітичних моделей які дають кращі результати ніж симулятори, але є більш трудомісткими в розробці та освоєні.

Крім того в даному розділі був проведений аналіз симуляторів накопичувачів, які є цінним інструментом для розроблення нових політик управління накопичувачами в режимах енергозбереження.

Також було розглянуто моделі «чорних скриньок» для накопичувачів, які просто будуються та не дуже залежать від показників які вклав виробник в пристрій, але за те їх легше пристосувати до нових моделей накопичувачів.

Ще одним з напрямків які було проаналізовано в цьому розділі – це енергозберігаючі рішення для накопичувачів які дозволяють менше споживати енергії, наприклад енергоефективні методи кешування та попередньої обробки, зовнішнього кешування, методів міграції даних між накопичувачами, що є актуальним для мобільних платформ, щоб як найдовше продовжити функціонування та дата центрів де неправильне управління або рішення може привести до значних фінансових втрат в загальному споживанні.

2 МОДЕЛЮВАННЯ ЧОРНОЇ СКРИНЬКИ ПОВЕДІНКИ ПРИСТРОЮ ЗБЕРІГАННЯ

2.1 Загальні принципи моделювання чорної скриньки

Моделювання є поширеним методом оцінки продуктивності багатьох додатків. Для отримання реалістичних результатів не можна ігнорувати доступ до даних у файловій системі або у системі управління базами даних. Однак у будь-якому випадку ключовим елементом доступу до даних є імітаційна модель пристрою зберігання даних.

Загалом, модель пристрою зберігання даних приймає на вході параметри робочого навантаження додатку (як потік запитів до пристрою) і виводить прогноз метрики продуктивності. Вихідним виміром продуктивності може бути загальна метрика продуктивності, наприклад, середня пропускна здатність, продуктивність або затримка. Такі показники дають уявлення про глобальну продуктивність пристрою. Однак, якщо імітаційна модель пристрою має бути інтегрована в більшу модель, яка включає файлову систему або менеджер баз даних, необхідна детальна метрика, наприклад, час відгуку на кожен запит пристрою.

Традиційно пристрої зберігання даних моделювалися за допомогою детальних аналітичних моделей, заснованих на геометрії пристроїв та їх схетотехнічним рішенням [65], розділенні зон [66] або використанні кешів випередження читання і переупорядкування запитів [67], що в багатьох випадках досягало рівня емуляції. Одним з недоліків аналітичних моделей є те, що як тільки з'являється нова технологічна інновація, модель потрібно переробляти для включення нових пристроїв, що ускладнює підтримку загальної моделі в актуальному стані.

Багато з цих недоліків були враховані в інструментах імітаційного моделювання. Одним з найпоширеніших інструментів моделювання, в якому інтегровано багато з цих ідей, є DiskSim. Симулятор використовує багато параметрів, які витягуються з дисків за допомогою напівавтоматизованих

алгоритмів [68], або за допомогою інструменту для визначення характеристик дисків DIXtrac. Щоб дати уявлення про складність моделі, достатньо зазначити, що DIXtract виокремлює понад 100 критичних параметрів продуктивності.

Існують загальносистемні моделювання, де масштабованість є важливим питанням. Це стосується проектування великих кластерів [69], оцінки однорангових або багаторангових обчислювальних моделей, інфраструктури зберігання даних Grid або мереж доставки контенту [70]. У всіх цих випадках реалістична симуляція повинна бути запущена протягом тривалого часу моделювання. Для того, щоб імітаційний підхід був доступним, необхідна модель пристрою зберігання даних з балансом між точністю та продуктивністю.

Альтернативним підходом є моделі "чорної скриньки", які не містять жодних знань про пристрій зберігання даних. Найпростішими моделями чорної скриньки є табличні моделі [71], хоча існують і більш складні моделі [72].

Використовуючи підхід "чорної скриньки", поведінку пристрою зберігання можна змоделювати за допомогою послідовності випадкових величин T_i, яка моделює час, необхідний для обслуговування і-го запиту. Ця послідовність випадкових величин є стохастичним процесом. Мета імітаційної моделі полягає в тому, щоб генерувати значення, які відповідають цьому стохастичному процесу. Для отримання такої імітаційної моделі необхідно отримати та проаналізувати експериментальні дані, щоб знайти розподіл, який лежить в її основі.

Однак справжньою причиною знаходження розподілу за експериментальними даними є можливість генерувати випадкові величини відповідно до цього розподілу. Другим підходом є генерація таких значень або з гістограми експериментальних даних, або з представлення кумулятивної функції розподілу експериментальних даних.

2.2 Метод моделювання "чорної скриньки" для пристроїв зберігання даних

Пропонований метод, починається з послідовності дискових запитів (від синтетичного робочого навантаження або реальних репозиторіїв трас) і закінчується генератором змінних, здатним створити кілька екземплярів імітованих трас часу обслуговування. Метод складається з декількох кроків, як показано на рисунку 2.1:

1.- Отримання послідовностей запитів на ввід/вивід

2 - Вимірювання часу обслуговування

3 - Побудова генератора змінних

4 - Отримання результатів моделювання

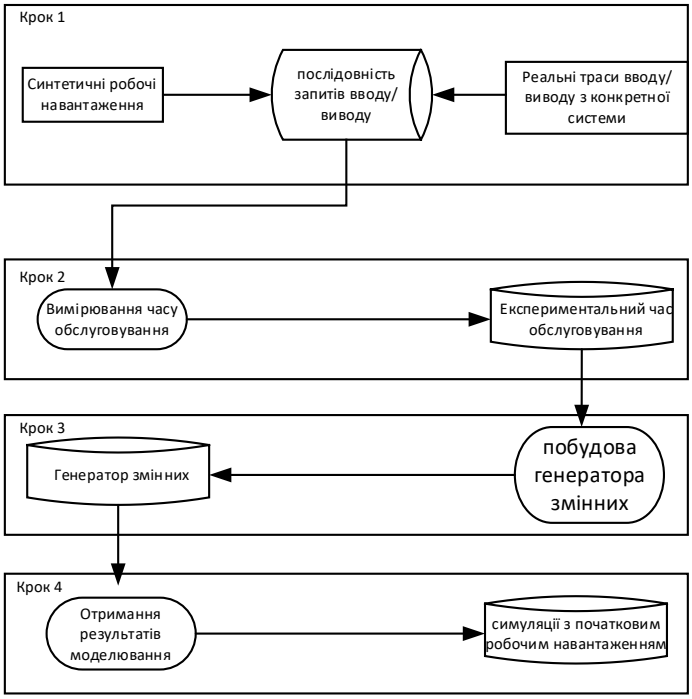


Рисунок 2.1 - Етапи моделювання "чорної скриньки" пристроїв зберігання даних

Перший набір даних (крок 1) - це послідовність запитів вводу/виводу, яка використовується для вимірювання часу обслуговування диска. Це навантаження може надходити з двох різних джерел:

- Реальні траси вводу/виводу з конкретної системи. Перевагою такого підходу є його реалістичність. Однак, з точки зору стохастичного моделювання, він не дозволяє запускати різні реалізації експерименту з різними вхідними даними.

- Синтетичні робочі навантаження, що моделюють набір реалізацій трас вводу/виводу. Цей підхід може здатися не таким реалістичним, але він легко дозволить запустити різні реалізації експерименту з різними вхідними даними.

Вибір джерела для запитів вводу/виводу залежить від призначення симулятора, що створюється. Пропонований метод дозволяє працювати, як з реальними трасами вводу/виводу, так і з синтетичними робочими навантаженнями, якщо використовується спільне представлення. Важливо зазначити, що метод чутливий і залежить від наборів даних запитів вводу/виводу.

Якщо використовувати набори даних про запити вводу/виводу, що надходять з веб-сервера, згенерований модуль буде точно імітувати час обслуговування диска за цих умов навантаження, але не за умов навантаження, що надається сервером бази даних.

Наступним кроком (крок 2) є отримання наборів експериментальних трас з вимірами часу обслуговування диска. Для цього використовуються різні реалізації процедури вимірювання. У кожній реалізації надсилаються запити до диска і вимірюються час обслуговування кожного окремого запиту, записуючи всю інформацію в репозиторій експериментальних трас.

Після експериментального вимірювання отримані дані необхідно проаналізувати для побудови генератора змінних (крок 3). Для цього можна використати два підходи - побудувати аналітичний або експериментальний генератор змінних:

– В аналатичному підході потрібно знайти розподіл за наборами даних.

– Для експериментального потрібен генератор змінних на основі гістограми або кумулятивного представлення функції розподілу і крім того він може бути легко автоматизований.

Генератор змінних включається в модуль моделювання (крок 4) для отримання результатів моделювання. Після запуску декількох реалізацій симуляції з початковим робочим навантаженням, використаним на кроці 2, результати симуляції порівнюються з експериментальними даними для визначення точності.

2.2.1 Отримання послідовностей запитів на ввід/вивід

Як було представлено в попередньому розділі, можна використовувати, як синтетичні, так і експериментальні траси. При використанні синтетичних трас обрано бенчмарк [73] для генерації трас, які є загальним представленням інших трас, і дослідники пробували отримати з них широкий спектр характеристик.

Використовуючи синтетичні траси, будуються більш загальні моделі, і хоча вони менш точні, їх можна використовувати для більшої кількості видів трас. З іншого боку, використовуючи реальні траси, можна побудувати більш специфічні моделі, здебільшого придатні для слідів, які також мають специфічні характеристики, що виявляються цими специфічними моделями. Крім того, специфічні моделі зазвичай є більш точними.

Для генерації синтетичних трас потрібно мати генератор синтетичних навантажень. Серед різних доступних варіантів які були обрані, як робоче навантаження, був SPC Benchmark v1.10.1 [73], визначений Storage Performance Council.

Бенчмарк SPC-1 визначає три блоки зберігання додатків (Application Storage Units, ASU), кожен з яких представляє логічний інтерфейс між сховищем даних і хост-програмами. Серед різних відображень ASU на логічний том, визначених SPC-1, вибирається N-1 map ping, в якому три ASU відображаються на один логічний том.

Поєднання трьох ASU, визначених у специфікації (сховище даних, сховище користувача та журнал), представляє типове використання системи зберігання даних. Кожен ASU складається з декількох потоків вводу/виводу (4

потоки для ASU1, 3 потоки для ASU2 і 1 потік для ASU3), параметри яких наведено в таблицях 2.1, 2.2 і 2.3.

Таблиця 2.1 - Application Storage Units1 Основні параметри

Параметр	1	2	3	4
Множник інтенсивності	0,035	0,281	0,07	0,21
частка читання	0,5	0,5	1	0,5
Адреса	рівном. розпод.	повторний випад. розп.	інкрементний доступ	повторний випад. розп.
Розмір	8	8		8

Таблиця 2.2 - Application Storage Units2 Основні параметри

Параметр	5	6	7
Множник інтенсивності	0,018	0,07	0,035
частка читання	0,3	0,3	1
Адреса	рівном. розпод.	повторний випад. розп.	інкрементний доступ
Розмір			

Таблиця 2.3 - Application Storage Units3 Основні параметри

Параметр	8
Множник інтенсивності	0,28
частка читання	0
Адреса	інкрементний доступ
Розмір	Змішаний

Для кожного потоку вводу/виводу SPC-1 визначає декілька параметрів. Основними параметрами є множник інтенсивності, що визначає вплив конкретного потоку в загальне робоче навантаження -це частка читання, що визначає кількість операцій читання (решта - операції запису), початкова адреса кожної операції та розмір запиту.

Адреси вирівнюються до 8 блоків з розміром блоку 512 байт. Розміри блоків виражаються у кількості 512-байтових блоків.

Як згадувалося раніше, схема доступу до кожного потоку сильно залежить від адреси та розміру передачі. Адреса може бути задана рівномірним

розподілом (Uniform), ієрархічним повторним випадковим розподілом [74] або інкрементним доступом (Inc).

Розмір запиту - це кількість 512-байтних блоків для читання або запису. Зазвичай цей параметр фіксується на рівні 8 блоків. Однак потоки 2, 7 і 8 використовують змішану модель, де розмір запиту варіюється від 8 блоків до 128 блоків.

Бенчмарк SPC-1 також визначає одиниці масштабування (Business Scaling Units, BSU), які представляють користувачів, що отримують доступ до трьох ASU. Кожна BSU складається з 8 потоків, описаних вище, і в цілому забезпечує навантаження в 50 операцій в секунду. Таким чином, кількість BSU прямо пропорційна навантаженню на диск, а коефіцієнт пропорційності дорівнює 50.

Для цього потрібен генератор. Мета генератора навантаження - отримати виміри для великої кількості запитів, щоб можна було побудувати генератор випадкових величин для диска.

Для досягнення цієї мети тривалість експериментів повинна бути достатньою для отримання точних оцінок. Використання таких тривалих експериментів накладає додаткову вимогу на базовий генератор випадкових чисел, якщо потрібно уникнути кореляції та автокореляції.

Щоб гарантувати, що різні джерела випадкових чисел базуються на генераторах випадкових чисел, які не перетинаються, використовується генератор випадкових чисел Мерсенна-Твістера [75], який має період, що дорівнює 219937^{-1}.

Для перевірки реалізації бенчмарку SPC-1 можна скористатись трьома потоками і не використовувати всі 8, оскільки їхні початкові адреси відповідають відповідно розподілам Uniform, RWalk та Inc. Решта потоків у інших ASU реалізують один з трьох дистрибутивів, але з різними параметрами.

Оскільки генератори випадкових чисел для даної реалізації та реалізації з відкритим вихідним кодом не однакові, просте використання однакових початкових даних може не покрити однакові адресні області для обох реалізацій. Тому потрібно використовувати однакові початкові адреси.

На рисунку 2.2 показано результати CDF (кумулятивної функції розподілу), отримані при виконанні (V-spc1) та відкритої (SD-spc1) реалізацій бенчмарку SPC-1 для змішаного розподілу за розміром. Вибірка складала 100 000 запитів і якщо проаналізувати вигляд цих графіків, то вони збігаються та корелюються.

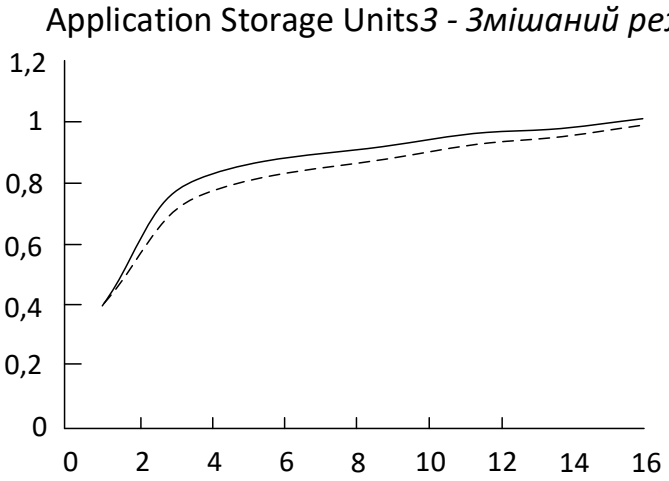


Рисунок 2.2 - Результати CDF запуску пропонованого та відкритої реалізацій бенчмарку для змішаного розподілу за розміром

На рисунку 2.3 показано результати CDF обох реалізацій тесту spc-1 для дистрибутивів Uniform (ASU1-1), RWalk (ASU1-2) та Inc (ASU1-3). Вибірки складалися з 10 мільйонів запитів для дистрибутивів Uniform та RWalk і 100 000 запитів для дистрибутива Inc.

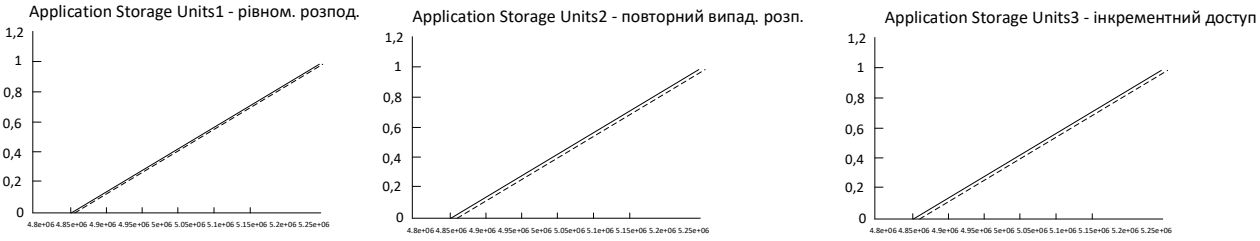


Рисунок 2.3 - Результати CDF обох реалізацій тесту spc-1 для розподілу стартових адрес Uniform (ASU1-1), RWalk (ASU1-2) та Inc (ASU1-3)

Початкові адреси були однаковими. Це призвело до того, що покрита територія також була однаковою, а CDF перекривалися. Як наслідок, на рисунку добре видно, що дистрибутиви дистрибутиви збігаються повністтю.

S3D - це програма для паралельного моделювання з використанням прямого чисельного моделюючого процесора. Контрольна точка виконується через регулярні проміжки часу, і її дані складаються в основному з розв'язаних змінних у 8-байтових тривимірних масивах, що відповідають значенням у точках тривимірної декартової сітки. S3D використовує колективний інтерфейс pnetCDF, але завдяки використанню підказок MPI-IO. Додаток виконує один мільйон 8-байтових записів, використовуючи незалежний режим вводу/виводу.

Бенчмарк NASA BTIO вирішує задачу Block-Tridiagonal (BT), яка використовує складну декомпозицію домену на квадратну кількість обчислювальних вузлів. Кожен обчислювальний вузол відповідає за декілька декартових підмножин всього набору даних. Під час виконання чергуються фази обчислень і фази вводу/виводу. Спочатку всі обчислювальні вузли спільно відкривають файл і оголошують види на відповідні області файлу. Після кожних п'яти обчислювальних кроків обчислювальні вузли записують розв'язок у файл за допомогою колективної операції. В кінці, отриманий файл колективно зчитується і розв'язок перевіряється на коректність [76].

2.2.2 Вимірювання часу обслуговування

Після того, як робоче навантаження доступне, наступним кроком є вимірювання часу відповіді для кожного запиту. Важливо зазначити, що метою тут є не отримання загальної метрики (наприклад, середнього часу відгуку), а отримання вимірів для кожного запиту як попереднє завдання для побудови імітаційної моделі. Робоче навантаження - це файл, який має стільки рядків, скільки запитів вводу/виводу. Кожен рядок містить інформацію про один запит, таку як адреса диска, на який він спрямований, його розмір, тип операції та часову мітку, коли запит повинен бути запущений. Отже, оцінювач

продуктивності отримує на вході попередньо отримане робоче навантаження. На виході генерується файл з часом відгуку.

Наскільки відомо, існує програма, яка досить добре справляється з цією задачею. Вона називається dxreplay і належить до пакету програм, який походить від інструменту DIXtrac. Маючи певне робоче навантаження, dxreplay виконує вимірювання часу відгуку на SCSI дисках і порівнює їх з вимірами, зробленими за допомогою Disksim. Це робиться для перевірки правильності вилучення параметрів, які відповідають детальній моделі реального диска у DiskSim. Він складається з головного потоку, який створює три екземпляри трьох потоків: lbnreader, issuer та collector. lbnreader читає запити з вхідного файлу робочого навантаження, issuer запускає раніше прочитані запити на диск у свій часовий мітка, а collector чекає на завершення запущених запитів. Всі три потоки виконуються по черзі і синхронізуються за допомогою м'ютексів. Однак, dxreplay працює лише з SCSI дисками. Тому ми реалізували play, нашу програму оцінки продуктивності, яка виконує вимірювання часу відгуку на будь-якому типі диска, а отже, моделює будь-який тип диска, з будь-яким інтерфейсом.

На відміну від dxreplay, play проводить вимірювання за допомогою стандартних викликів POSIX, а не команд SCSI. Він складається з головного потоку, який зчитує кожен запит від робочого навантаження і запускає його у вказаний момент часу (Алгоритм 1, рисунок 2.4). Він також містить обробник сигналів, який вмикається, коли певний запит завершився, і час його відповіді може бути переданий безпосередньо у вихідний файл, при виконанні налагоджувальних завдань або записаний у структуру даних в пам'яті (Алгоритм 2 рисунок 2.5).

Обслуговування декількох невиконаних запитів є поширеною функціональністю сучасних дисків. Саме тому наша реалізація оцінювача використовує асинхронний ввід/вивід. Кожна операція з робочого навантаження видається у вказаний час початку і для грубого вимірювання часу відгуку. Коли запит завершено, час обслуговування записується у структуру даних у пам'яті, яка скидається при завершенні програми. І play, і dxreplay генерують вихідні

файли, що містять час відгуку, які можна представити у вигляді CDF-файлів, як показано на рисунку 3.4

Для проведення вимірювань на даному накопичувачі з метою отримання часу обслуговування, ми змонтували диск як додатковий диск у системі, що має основний диск, на якому розміщується операційна система, інструменти оцінювання та пов'язані з ними файли з трасами оцінювання.

```
1. активація обробника сигналу
2. завантаження запиту в пам'ять з файлу робочого
навантаження
3. цикл (і <= до «кінця файлу»)
4.    {
5.     записати час запуску запитів для і-го кроку
6.     звантажити запит на диск за його міткою часу для
       і-го кроку
7.      чекати на наступний запит
8.    }
9. записати час відповіді у вихідний файл
вихід
```

Рисунок 2.4 - Алгоритм вимірювання за допомогою стандартних викликів POSIX

```
1. фіксувати час завершення запиту (завершений запит)
2. обчислити час відповіді (завершений запит)
3. вивести час відповіді за структурою даних  (завершений запит)
```

Рисунок 2.5 - Алгоритм обробника сигналів

Щоб уникнути будь-якого впливу на процедуру оцінювання з боку файлової системи та драйвера пристрою, а також для того, щоб гарантувати, що кожен запит на введення/виведення у робочому навантаженні фізично надсилається до оцінюваного диска, було використано операційну систему GNU/LINUX і ініціалізовано диск, як символьний пристрій, за допомогою команди mknod і зв'язавши його з диском, можна буде оцінювати його (за допомогою raw).

Таким чином, до пристрою можна отримати прямий доступ за допомогою викликів POSIX для читання і запису. Більше того, всі операції вводу/виводу

виконуються над адресним простором оцінюваного процесу за допомогою DMA, завдяки чому адреси на диску та в пам'яті вирівнюються до межі 512 байт.

Можливий шум від інших запущених процесів зведено до мінімуму шляхом запуску інструменту оцінки з мінімальною кількістю активних процесів (за допомогою режиму init 1).

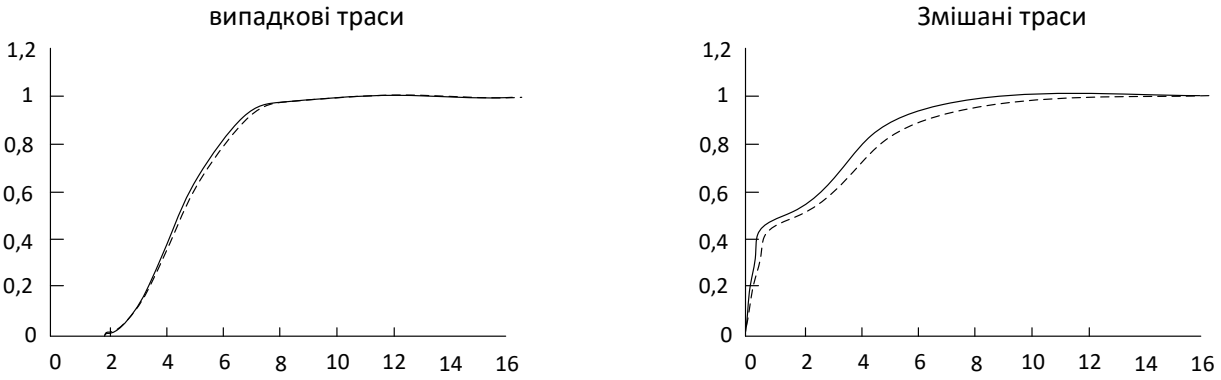


Рисунок 2.6 - CDF-графіки часу відгуку диска WD40EFAX та двох синтетичних трас. Час відгуку було отримано під час запуску команд play і dxreplay.

2.2.3 Побудова генератора змінних

Після вимірювання часу обслуговування можна «підігнати» отриманий раніше час відгуку до одного або декількох відомих розподілів, щоб побудувати генератор змінних для таких розподілів. При побудові генератора змінних використовується вибірки часів відгуку. Метод побудови складається з декількох кроків, як показано на рисунку 2.7:

1.- Отримання часу відгуку

2.- Визначення статистичних розподілів

3 - Пристосування до статистичних розподілів

4 - Побудова моделі

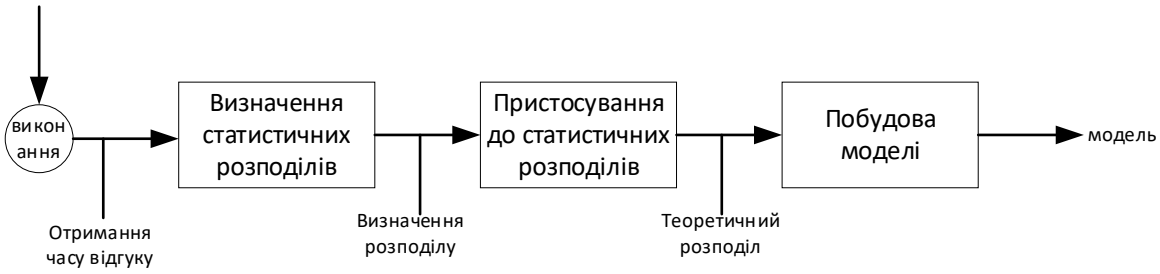


Рисунок 2.7 - Етапи процесу побудови генератора змінних

Перший набір даних (крок 1) - це вибірка часу відгуку, отримана при відтворенні траси на реальному диску. Ця вибірка походить від запуску play, програми вимірювання часу відгуку для певного робочого навантаження. Як було сказано раніше, навантаження може бути реальним або синтетичним.

Наступним кроком (крок 2) є виявлення статистичних розподілів з попередньо отриманої вибірки часу відгуку. Вибірка може містити кілька різних розподілів, які охоплюють різні діапазони часу відгуку. Кожен розподіл може бути результатом різних причин, які генерують час відгуку з одного або іншого розподілу. В якості причин можна назвати те, що конкретний запит може обслуговуватися з пластин диска або з його буферів, може чекати на обслуговування попередніх запитів і т.д.

Після виявлення можливих дистрибутивів, виявлені дистрибутиви необхідно зіставити з деякими відомими дистрибутивами (крок 3). Кожен окремий розподіл може бути єдиним розподілом або сумішшю розподілів. Незалежно від того, чи це поодинокий розподіл, чи суміш розподілів, використовувалось середовище статистичного аналізу R [77], щоб підігнати вибірки до теоретичних розподілів. При підборі одиночних розподілів застосовувався один метод, а при підборі змішаних - інший.

Підібрані розподіли використовуються для побудови моделі (крок 4). Пропонується два різних підходи: Один з них базується на реальних трасах, а інший - на синтетичних трасах. Підхід на основі реальних трас полягає в побудові окремих моделей для кількох реальних трас. Коли прогнозується час відгуку на основі конкретної траси, обирається одна з попередньо

змодельованих трас, і на її основі генерується час відгуку. З іншого боку, прогнозування на основі синтетичних трас полягає в побудові лише однієї моделі на основі загальної синтетичної траси. Коли час відгуку прогнозується на основі конкретної траси, модель здатна частково адаптуватися до траси для прогнозування.

2.2.4 Виявлення статистичних розподілів

Пропонований метод виявлення статистичних розподілів складається з наступних кроків:

1.- Представлення вибірки часу відгуку гістограмою

2.- Виявлення можливих розподілів

3.- Розбиття вибірки часу відгуку на виявлені розподіли

4.- Підгонка виявлених розподілів до теоретичних розподілів. Якщо розподіли не можуть бути підігнані, повернення до кроку 1.

При виявленні статистичних розподілів першим кроком є представлення попередньо отриманих даних у вигляді гістограми. Гістограма дає зображення розподілів, які відповідають раніше отриманим даним. Кожен конкретний розподіл представляє один або декілька варіантів поведінки накопичувача за певних характеристик. Наприклад, один з розподілів може відображати час відгуку, який генерують дані, розміщені у кеші диска. Інший дистрибутив може показувати час відгуку від даних, які не зберігаються у кеші накопичувача. Інші дистрибутиви можуть ілюструвати час відгуку від робочих навантажень, у яких накопичувач простоює протягом певних періодів часу. Деякі розподіли можуть показувати час відгуку від дуже великих навантажень.

При виявленні дистрибутивів виникає ситуація коли два різні дистрибутиви досить сильно розрізняються за діапазоном своїх доменів, і їх легко розділити. Деколи вони можуть бути дуже близько, і тоді стає важко визначити, де їх розподіл. У таких випадках робиться розподіл за спільним значенням мвсця розподілу.

Після розподілу можна переходити до «підгонки» виявлених індивідуальних вибірок під теоретичний розподіл. Якщо вони легко підходять, і якість підгонки задовільна, то вважається, що компоненти вибірки ідентифіковані. В іншому випадку знову представляється початковий розподіл гістограми і повторно виявляються компоненти вибірки та величина об'єднання.

На рисунку 2.8 показано форму гістограми для диска WD40EFAX. На гістограмі можна виділити два розподіли:

розподіл 1, який представляє час відгуку, згенерований нормальною активністю диска;

розподіл 2, який представляє час відгуку, згенерований, коли диск попередньо простоював, протягом певного періоду часу.

У цьому випадку компоненти вибірок не були близькими, і значення об'єднання було легко визначити. Крок становив 30 мс. Після підбору компонентів вибірки і перевірки їхньої відповідності, вважається що розподіл правильний і алгоритм виявлення не потрібно повторювати.

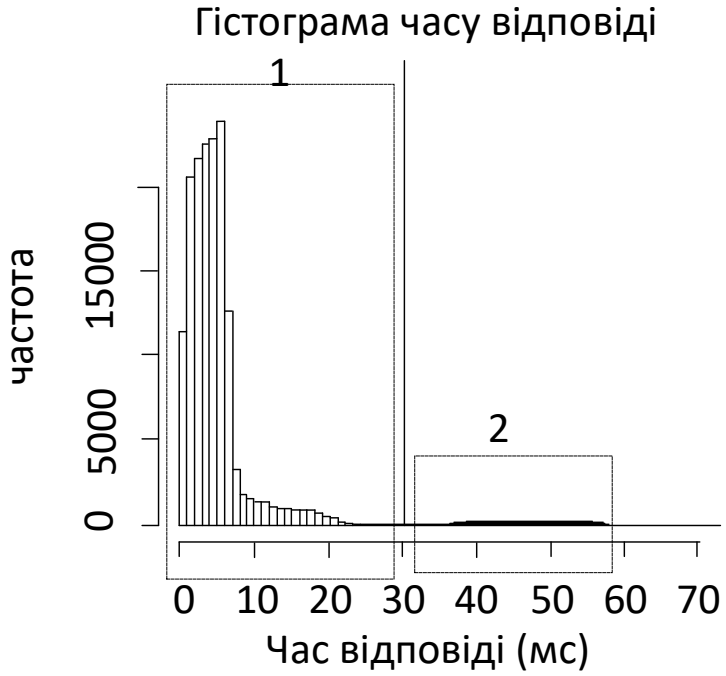


Рисунок 2.8 - Частини розподілу, які представляють час обслуговування з диска WD40EFAX

2.2.5 Підгонка до статистичних розподілів

Метод підгоники до статистичних розподілів складається з кількох кроків:

1.- Представлення компонента вибірки гістограмою.

2.- Визначення, чи є компонент вибірки сумішшю розподілів.

3.- Якщо компонент вибірки не є сумішшю, приведення його до одного теоретичного розподілу

4.- Якщо компонент вибірки є сумішшю, підігнати його до теоретичної суміші розподілів.

Для того, щоб підігнати набір даних до невідомого розподілу, першим кроком є перевірка його специфічної гістограми, щоб визначити, чи не є експериментальні дані сумішшю розподілів. Якщо суміш не виявлено, виконується функція fitdistr з середовища статистичного аналізу R [77]. Аргументами цієї функції є експериментальні дані та розподіл, до якого потрібно підігнати дані. fitdistr виконується один раз для кожного можливого розподілу (нормального, гамма, експоненціального) і повертає оцінки параметрів для такого розподілу. За допомогою оцінок параметрів за допомогою статистики розбіжностей λ 2 [78] кількісно оцінюється відповідність усіх можливих розподілів і вибирається розподіл з найкращою відповідністю

Якщо розпізнано суміш розподілів, використовується пакет mixdistr із середовища статистичного аналізу R [77]. Для підгонки суміші першим кроком є групування експериментальних даних. Для цього виконується функція mixgroup. Функція групує експериментальні дані у вигляді чисел спостережень за відповідними інтервалами. Аргументами є експериментальні дані та кількість інтервалів. У цьому випадку знаходиться кількість інтервалів за допомогою функції binning, яка автоматично їх обчислює. Наступним кроком є надання додаткової інформації про форму та параметри експериментальних даних. Це робиться за допомогою перегляду гістограми експериментальних даних. Спочатку визначаються початкові значення для середніх та сигм і вирівнюються пропорції. Потім оцінюються параметри суміші, виконуючи функцію mix.

Аргументами для цієї функції є попередньо згруповані дані, оцінені параметри, компоненти сумішей (нормальна, лог-нормальна, експоненціальна, гамма, Вейбулла, біноміальна, від'ємно біноміальна та пуассонівська), обмеження та метод/алгоритми, які потрібно використати. Зазвичай розглядається постійний коефіцієнт варіації, як початкове обмеження і використовується комбінація алгоритму EM [79] та методу Ньютона. При використанні процедури EM необхідно передбачити кількість кроків – 3 кроки було цілком достатньо.

У прикладі з диском WD40EFAX за трасою [80] обидва частково виявлені розподіли є сумішами. Зокрема, обидва розподіли є сумішшю двох нормальних розподілів, параметри яких наведено в таблиці 2.4. Їхні форми також показано на рисунку 2.9.

Таблиця 2.4 - Параметри ймовірнісних розподілів, що моделюють час роботи диска WD40EFAX

	тип	π	μ	σ
Розподіл 1	Нормальний	0,89	3,73	1,96
	Нормальний	0,11	10,82	5,71
Розподіл 2	Нормальний	0,92	47,04	6,38
	Нормальний	0,08	29,42	3,99

Після підбору розподілів до експериментальних даних видно, наскільки правильно були підібрані параметри, побудуваної гістограми для експериментальних даних з розрахованими розподілами. На рисунку 2.9 наведено приклад такої візуалізації. Також використовувались інші візуальні методи, щоб оцінити відповідність певного розподілу експериментальних даних, наприклад, Q-Q-графік (рисунок. 2.10 - ліворуч). На діаграмі Q-Q чим більше пунктирна лінія наближається до діагоналі, тим кращою є відповідність розподілу, оскільки тим кращою є відповідність між квантилями.

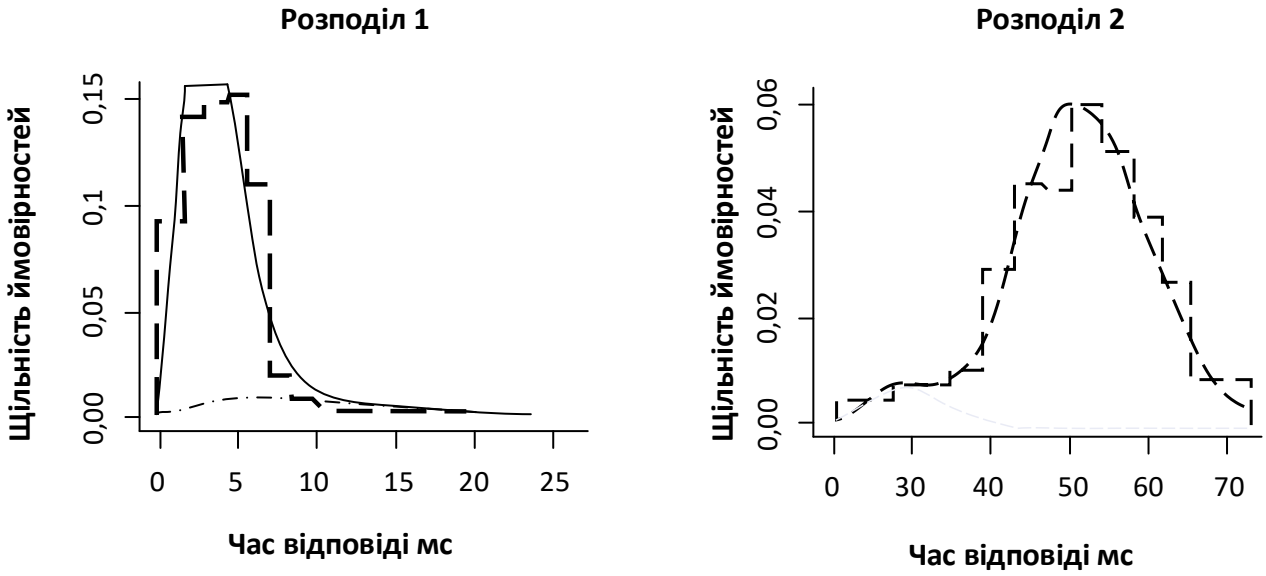


Рисунок 2.9 - Порівняння двох гістограм, побудованих на основі експериментальних даних, та розподілів, до яких вони підходять

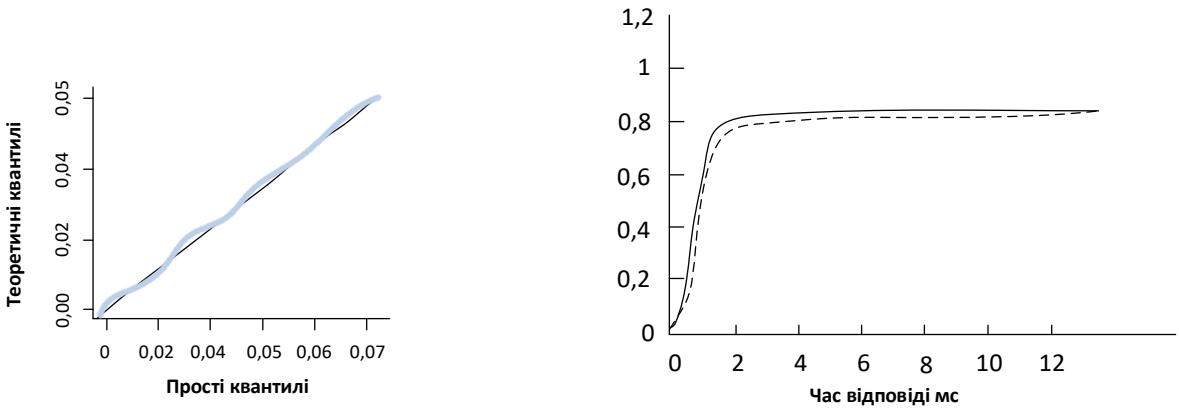


Рисунок 2.10 - Два візуальні методи, що дозволяють побачити, наскільки добре змодельовані розподіли відповідають реальним розподілам.

На графіку ліворуч - порівняння квантилів обох розподілів. Накладання CDFs дозволяє візуалізувати, наскільки добре розподіли відповідають один одному – правий графік.

Серед різних числових вимірювань використовувалось рівняння [81]. Воно визначається, як середній корінь квадратний з горизонтальних відстаней між прогнозованими та реальними функціями розподілу часу відгуку. Також

зображено прогнозовані та реальні дані, щоб побачити, наскільки добре вони накладаються (Рисунок 2.10 - праворуч).

Наступним кроком є створення модуля моделювання за допомогою середовища моделювання OM NET++ [82]. Модуль просто містить один або декілька генераторів випадкових величин, які генерують імітаційні часи відгуку на основі попередньо підібраних функцій густини ймовірності.

Щоб гарантувати, що різні джерела випадкових величин базуються на генераторах випадкових чисел, які не перетинаються, було використано генератор випадкових чисел Мерсенна-Твістера [83], який має період, що дорівнює $2^{19937 - 1}$.

2.3 Побудова моделі

Процес побудови моделі запропонованого методу складається з наступних кроків:

1.- Вибір типу трас, на яких будується модель.

2.- Якщо траси є реальними, побудова моделі для кожного з ідентифікованих та підігнаних розподілів цих реальних трас

3) якщо траса є синтетичною, побудова моделі для синтетичної траси без урахування часу опитування.

З іншого боку, прогнозування часу відгуку на основі попередньо побудованих моделей складається з додаткових кроків:

1.- Вибір типу трас, на яких будується модель.

2.- Якщо траси є реальними, прогнозування на основі однієї з змодельованих реальних трас, шляхом визначення тієї, яка має найбільш схожі характеристики з трасою для прогнозування.

3) Якщо траса є синтетичною, прогнозування на основі синтетичної змодельованої траси та обчислення часу запиту "на льоту".

При побудові моделей визначається причини попередньо виявлених і підігнаних до теоретичних розподілів вибірок. Під причинами розуміються

характеристики навантаження, які призводять до того, що час відгуку відповідає певному розподілу, а не іншим. Серед кількох причин можна виділити наступні:

- Тривалі періоди бездіяльності. Коли накопичувач простоює певний час, наступний запит, який потрібно обслужити, може тривати довше, ніж зазвичай.

- Час очікування в черзі. Коли запит на обслуговування надходить, коли один або декілька попередніх запитів ще не завершили свою роботу, він повинен дочекатися завершення попередніх запитів, що збільшує час його відповіді.

- Послідовність. Коли кілька запитів є послідовними, різниця в LBN між ними незначна. І навпаки, коли різниця в LBN велика, послідовність може бути не визначена, і час відповіді буде більшим.

- Ефекти кешування. Накопичувачі мають внутрішні буфери. Коли їх використання активоване, обслуговування запитів з буферів накопичувача відбувається швидше, ніж обслуговування запитів безпосередньо з пластин.

Отже, після того, визначено причини генерування часу відгуку для кожного конкретного розподілу, будується модель, в якій, залежно від вхідних даних, вибирається той чи інший теоретичний розподіл для генерування часу відгуку.

На прикладі диска WD40EFAX, під слідом [uma11], визначаються два розподіли і три причини для них. Як ми вже згадувалось раніше, розподіл 1 є сумішшю двох нормальних розподілів (див. Таблицю 2.4). Час відгуку з частини 1 розподілу 1 генерується, коли запити повинні обслуговуватися з дисків. Час відгуку з частини 2 розподілу 1 генерується, коли періоди бездіяльності довші за 1 секунду і коли запити, що обслуговуються з дисків, повинні чекати на завершення попередніх і час відгуку з розподілу 2 генерується, коли періоди бездіяльності становлять близько півсекунди. Отже, для цієї конкретної моделі розроблено алгоритм, який показано на рисунку 2.11

```
if   ((simTime   ()-init_disk   [actReq   -   1])>520)
&((simTime()-ini_disk[actReq-1]) <700) then
  choose ⟵ Bernoulli (0.92208)
  if choose =1 then
    response_time⟵ normal (47.04, 6.385)
  else
    response_time⟵ normal (29.42, 3.993)
  end if
else
  if   ((simTime   ()-init_disk   [actReq   -   1])>1000)
  ||(simTime()-end_disk[actReq-1])
  then
    response_time ⟵ normal (10.819, 5.706)
  else
    response_time ⟵ normal (3.727,1.965)
  end if
end if
```

Рисунок 2.11 - Алгоритм представлення моделі

Генерація часу відгуку від розподілу 2 описана у рядках 1-8. Кожного разу, коли накопичувач простоює більше 520 мс і менше 700 мс, фактичний запит actReq генерує час відгуку від однієї з частин розподілу 2. Відстеження цього накопичувача шляхом віднімання відмітки часу надходження actReq (simT ime()) від відмітки часу надходження попереднього запиту (ini disk[actReq-1]). Ймовірність вибору частини 1 з дистрибутиву 2 вища (0.92208), ніж вибору частини 2. Визначення частини використання проводиться за допомогою розподілу Бернуллі (рядок 2).

Генерування часу відгуку за розподілом 1 описано в рядках 9-14. Кожного разу, коли диск простоює більше 1000 мс, фактичний запит actReq генерує свій час відгуку з частини 2 розподілу 1 (рядок 10). Також, якщо actReq надходить (simT ime()), а попередній запит actReq-1 ще не було обслуговано, час відповіді генерується з частини 2 Розподілу 1 (рядок 10). Таким чином, імітується, що actReq було поставлено в чергу. У будь-якому іншому випадку час відповіді формується з частини 1 Розподілу 1 (рядок 12).

Як показано в оцінці, активація використання внутрішніх буферів призведе до збільшення часу відгуку з іншого розподілу, коли запити мають обслуговуватися з буферів диска.

2.3.1 Побудова моделей на основі реальних трас

При побудові моделей на основі реальних трас, для кожної реальної траси виконуються ті самі кроки, що описані в попередньому розділі. Отже, для кожної реальної траси виділяються декілька причин, щоб згенерувати час відгуку з одного підібраного розподілу або з інших, якщо існує більше одного розподілу. Якщо існує лише один розподіл, то всі часи відгуку генеруються з нього.

В кінцевому результаті отримується пул моделей з декількох реальних слідів з різними характеристиками. Прогнозування на основі цієї моделі передбачає вибір однієї з декількох попередньо змодельованих трас. Критерії вибору траси базуються на середньому розмірі запиту та середньому часі очікування в черзі траси для моделювання. Також береться до уваги послідовність.

На основі згаданих критеріїв вибирається попередньо змодельована траса і використовується для прогнозування часу відгуку від вхідного навантаження.

Час від часу критерії перевіряються знову, і якщо вони змінилися, вибирається інша найбільш схожа раніше змодельована траса. Для того, щоб цей підхід був точним, має бути принаймні одна раніше змодельована траса зі схожими характеристиками, як і траса, яку потрібно спрогнозувати (рисунок 2.12) показує, як працює прогнозування в цій моделі:

```
1.    if (contLastReqs < LastReqs) than
2.      if (distr = 0) then
3.        response_time ←— generate S3D ()
4.      end if
5.      if (distr = 1) then
6.        response_time ←— generate BTIO ()
7.      end if
8.      if (distr = 2) then
9.        response_time ←— generate ModBench()
10.     end if
11.     if (distr = 3) then
12.       response_time ←— generate Fin()
13.     end if
14.     if (distr = 4) then
15.     response_time ←— generate Cello99()
16.     end if
17.     countLastReqs++
18.     countLastSizeToStatistics ()
19.     addQueuingTimeToStatistics ()
20.     addSwquentalityToStatistics()
21.     else
22.     calculateDistr ()
23.     contLastReqs  ←— 0
24.     end if
```

Рисунок 2.12 - Алгоритм прогнозування

Перед надходженням першого запиту contLastReqs ініціалізується 0. lastReqs - це константа, яка визначає максимальну кількість запитів, що надійшли, перед тим, як перевірити, чи потрібно вибрати інший розподіл.

Якщо дистрибутив потрібно вибрати, виконується метод calculateDistr() (рядок 22). Цей метод на основі деяких критеріїв визначає, який розподіл є найбільш придатним для прогнозування вхідної траси. Відповідно до обраного розподілу, значення якого знаходиться у змінній distr, з нього генеруються часи відгуку (рядки 2-16). Кожного разу, коли потрібно згенерувати час відгуку для конкретного запиту, статистика вхідного навантаження також має бути оновлена. Ця статистика включає середній розмір запиту (addReqSizeToStatistics(), рядок 18), середній час очікування у черзі (addQueuingTimeToStatistics(), рядок 19) та послідовність (addSequentialityToStatistics(), рядок 20).

2.3.2 Побудова моделей на основі синтетичних трас

При побудові моделей на основі синтетичної траси також всі кроки виконуються і для синтетичної траси, Для цієї побудови виділяється декілька причин для вибору між одним придатним розподілом та іншими, якщо їх існує декілька.

Головна відмінність від побудови моделей на основі реальних трас полягає в тому, що, крім того, що використовується лише одна траса, час очікування в черзі не моделюється в отриманому часі відгуку.

Це можна зробити, використовуючи характеристику відтворення, програми вимірювання часу відгуку. Як було сказано раніше, програма дозволяє користувачеві отримати час відгуку, уникаючи часу очікування в черзі. Це можна зробити, обмеживши максимальну кількість запитів вводу/виводу, які можуть бути поставлені в чергу на диск, до 1.

Оскільки час очікування в черзі є одним з найбільш значущих і характерних ефектів у часі відгуку, то кожна траса вводу прогнозує його самостійно, моделі можуть бути більш універсальними і загальними. В

алгоритмі на рисунку 2.13 показано, як для попередньо побудованої моделі накопичувача WD40EFAX розрахувати час очікування в черзі "на льоту".

```
1.        if((simTime()-ini_disk[actReq-1])>0.52)&((simTime()-ini_disk[actReq-
2.        1]) <0.7) then
3.            choose <= Bernoulli (0.92208)
4.            if choose = 1 then
5.              response_time <= normal (47.04, 6.385)
6.            else
7.              response_time <= normal (29.42, 3.993)
8.            end if
9.        else
10.           if ((simTime()-ini_disk[actReq-1])>1 then
11.             response_time <= normal (10.819, 5.706)
12.           else
13.             response_time <= normal (3.727, 1.965)
14.           end if
15.       end if
16.       if ((simTime()-end_disk[actReq-1]) then
17.       response_time <= Response_time+ (end_disk[actReq-1]-simTime ())
18.       end if
```

Рисунок 2.13 – Алгоритм розрахунку часу «на льоту»

Хоча обрана траса не є синтетичною, метою було показати різницю між генеруванням часу очікування на основі попередньо змодельованого розподілу та при генеруванні на льоту.

Як було сказано раніше, генерація часу відгуку з розподілу 2 рядки 1 - 8. Кожного разу, коли диск простоює більше 520 мс і менше 700 мс, фактичний запит actReq генерує свій час відповіді з однієї з частин розподілу 2.

Час простою визначається шляхом віднімання відмітки часу надходження actReq (simT ime()) від відмітки часу надходження попереднього запиту (ini disk[actReq - 1]). Ймовірність вибору частини 1 з дистрибутиву 2 вища (0.92208), ніж вибору частини 2. Визначення, яку частину використовувати, за виконується за допомогою розподілу Бернуллі (рядок 2).

Генерування часу відгуку за розподілом 1 описано в рядках 9-14. Кожного разу, коли диск простоює більше 1000 мс, фактичний запит actReq генерує свій час відгуку з частини 2 розподілу 1 (рядок 10). У всіх інших випадках час відгуку генерується з частини 1 розподілу 1 (рядок 12). Якщо actReq надходить (simTime()), а попередній запит actReq-1 ще не було обслуговано, actReq ставиться в чергу. Щоб симулювати, що він був поставлений у чергу, обчислюється час його відповіді, додаючи час його перебування у черзі (end

disk[actReq - 1] - simT ime() - рядок 16) до попередньо обчисленого часу обслуговування (часу відповіді) .

Висновки до розділу 2:

У цьому розділі описано метод побудови моделі чорної скриньки для жорстких дисків. Він базується на розподілі ймовірностей і може бути використаний, як для синтетичних, так і для реальних рефлектограм. Метод включає в себе реалізацію вимірювання часу обслуговування, яка може бути використана в кожному диску, з будь-яким типом інтерфейсу.

Для побудови моделей запропоновано два підходи. Перший підхід базується на декількох реальних трасах. Він може бути точним, коли траси для прогнозування мають схожі характеристики з однією з трас, для якої була побудована модель. Другий підхід базується на одній синтетичній трасі. Передбачається, що ця траса охоплює широкий спектр характеристик інших трас. Цей підхід обчислює час очікування в черзі на льоту, на етапі прогнозування, що робить його більш універсальним.

Даний метод забезпечує точний і більш загальний альтернативний підхід для моделювання часу обслуговування будь-якого пристрою, з будь-яким типом інтерфейсу, тому, що, на відміну від інших підходів, його реалізація вимірювання часу обслуговування використовує стандартні виклики POSIX. Крім того, оскільки використовується модель "чорного ящика", багато характеристик пристрою не потрібно знати, що робить його простішим, менш детальним і, отже, легшим.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МЕТОДУ “ЧОРНОЇ СКРИНЬКИ”, ДЛЯ ЕНЕРГООЩАДНОЇ АРХІТЕКТУРИ ДЛЯ НАКОПИЧУВАЧІВ

3.1 Оцінка моделювання "чорної скриньки

Метод був протестований, використовуючи накопичувач WD40EFAX-68JH4N0, DCM , Western Digital 4TB SATA 3.5 Hard Drive [84], як бета-тест. Його основні характеристики наведено в таблиці 3.1. Було побудовано кілька моделей чорної скриньки для диска і перевірено моделі, використовуючи сценарії, як були представлені в попередніх розділах.

Таблиця 3.1 - Параметри ймовірнісних розподілів, що моделюють час роботи диска WD40EFAX

Специфікація	Параметр
Головки	4
Диски	2
Користувацькі сектори на диск	5 860 533 168
Швидкість	5400
Швидкість передачі інтерфейсу	150 MB/s
Цикли завантаження/розвантаження	600000
Кеш (Мб)	64

Для управління системою був вибраний дистрибутив Debian GNU/Linux де було написано основні сценарії реалізації алгоритмів роботи з накопичувачем а також середовище LabView [85] для побудови моделей, візуалізації результатів та роботи по обміну даними між Linux машиною де проводились тестування і Windows системою де проводився аналіз роботи.

У даній реалізації порівнюються вимірювання часу обслуговування, play, з dxreplay, інструментом вимірювання часу обслуговування, що входить до складу DIXtrac. Порівнюються обидві програми, через надсилання їм синтетичних траси в різних конфігураціях.

На рисунку 3.1 показано результати CDF (кумулятивної функції розподілу) запуску двох синтетичних трас на диску WD40EFAX за допомогою

програми dxreplay з DIXtrac, порівняно з результатами запуску на тому ж диску за допомогою програми play вимірювання часу обслуговування.

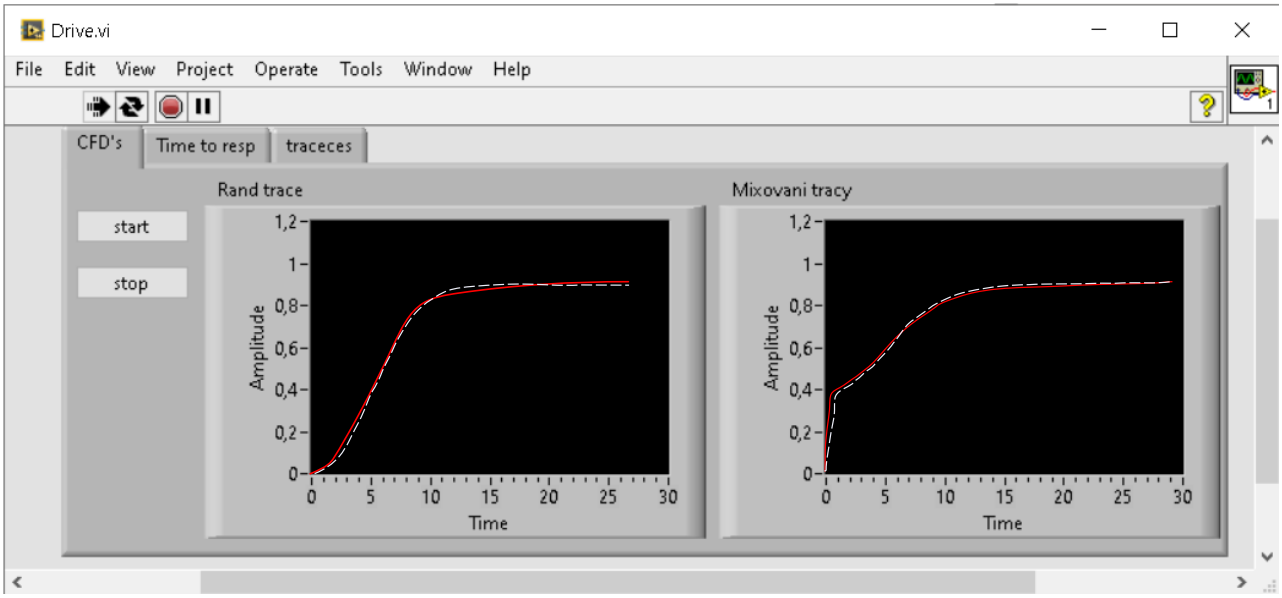


Рисунок 3.1- CDF для часу відгуку диска WD40EFAX і двох синтетичних трас.

Штрих-пунктирною лінією показано роботу запропонованої програми, а суцільною – пакет DIXtrac.

Траса, Random, містить 10 000 запитів, з яких 233 - читання і 13 - запис. LBN випадкові і розподілені по всьому диску. Розмір запитів коливається від 1 КБ до 8 КБ. Інша траса, змішана, містить 5,000 запитів; 23% від загальної кількості запитів - на читання, а решта - на запис і 20% запитів є послідовними, а 30% - локальними, решта 50% мають випадкові LBN. Розмір запитів коливається між 1 КБ та 8 КБ.

Було використано помилку [86], як метрику для перевірки програми контролю часу обслуговування. Вона визначається, як середній корінь квадратний з горизонтальної відстані між розподілом, отриманим від dxreplay, і розподілом, отриманим від створеної програми вимірювання часу обслуговування, play. Вона представляється в абсолютному вираженні (як різниця в мілісекундах), щоб порівняти його з іншими моделями DIXtrac.

Випадковий запит виконувався після деактивації буферного кешу на диску. Mixed було виконано після повторного увімкнення кешу, а отже, звітів про читання з випередженням і негайний запис.

Показники затримки, отримані для трас Random і Mixed, становлять, відповідно, 0,59 мс і 0,47 мс. Це досить хороший результат, тому його було використано у решті оцінок.

Dxreplay включає механізм обмеження максимальної кількості очікуваних запитів на введення/виведення, які можуть бути поставлені в чергу на диску, через ефект сплеску робочого навантаження. Тому такий самий механізм було включено у розроблену програму вимірювання часу обслуговування play, використовуючи м'ютекси.

Після чого знову було проведено порівняння обидвох інструментів вимірювання, виконавши раніше описані синтетичні траси, але зробивши їх більш перевантаженими. Щоб побачити ефект механізму обмеження черги, було виконано обидві траси, дозволивши 1, 2 та 3 запити з максимальним часом очікування в черзі.

На рисунку 3.2 показано CDF результати запуску нової траси Mixed на диску WD40EFAX за допомогою dxreplay, порівняно з результатами запуску на тому ж диску за допомогою play. Кеш диска було активовано. Показники часу, отримані для 1, 2 і 3 максимальних запитів у черзі, становлять 1.3 мс, 3.1 мс і 1.4 мс, відповідно.

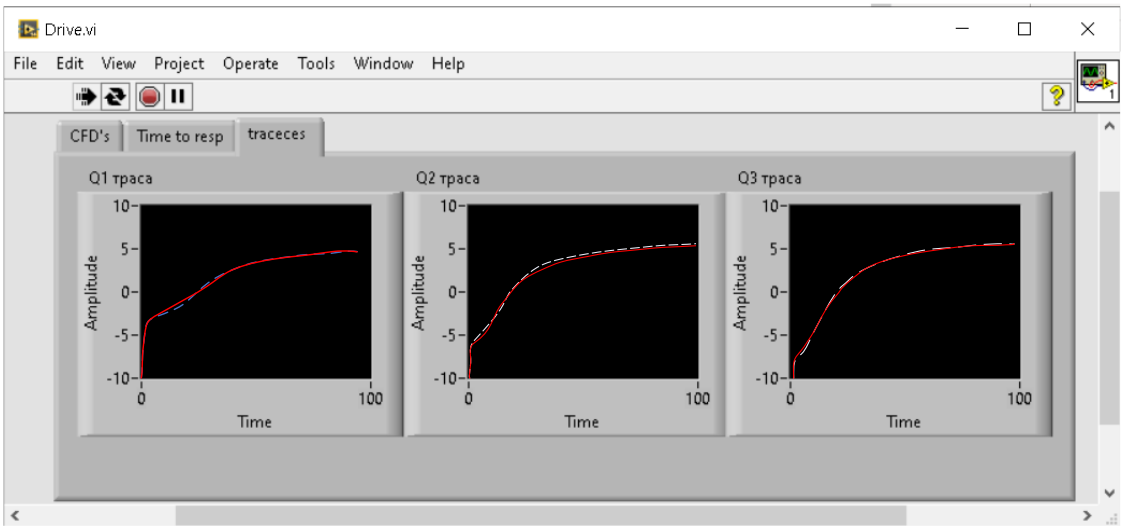


Рисунок 3.2 - CDF для часу відгуку від диска WD40EFAX та змішаної траси

Максимальна кількість запитів на введення/виведення, що очікують відповіді, які можна поставити в чергу на диску, обмежена 1 (Q-1), 2 (Q-2) і 3 (Q-3).

На рисунку 3.3 показано CDF результати запуску нової траси Random з розривом на диску WD40EFAX за допомогою dxreplay, порівняно з результатами запуску на тому ж диску за допомогою за допомогою play. Кеш на диску було деактивовано.

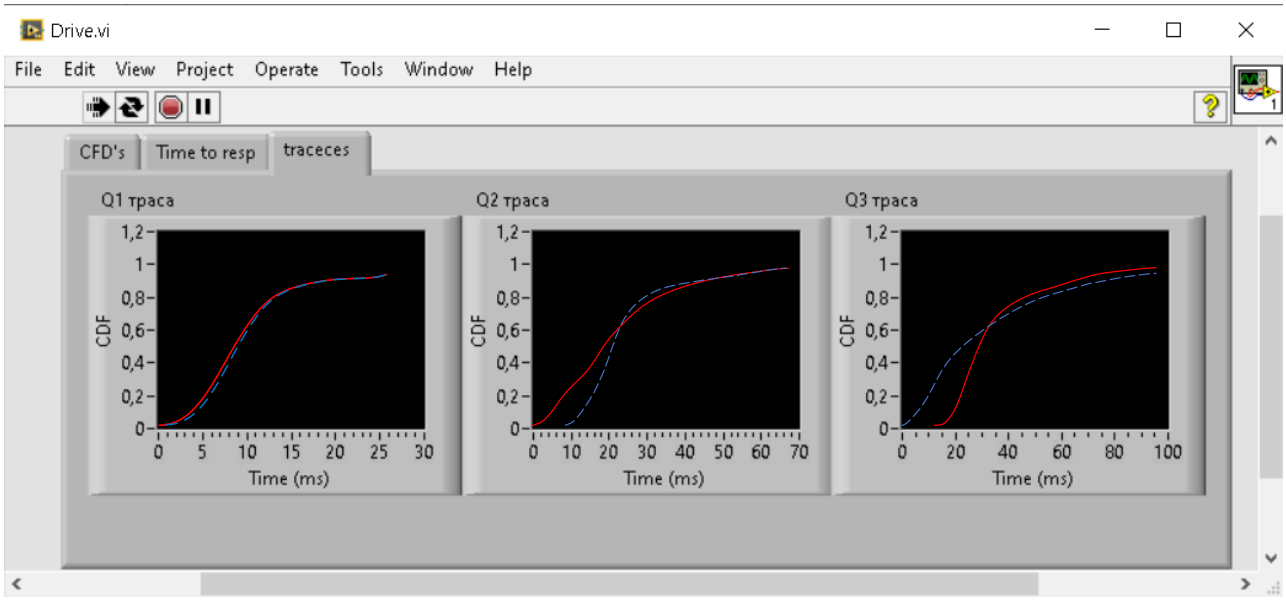


Рисунок 3.3 - CDF результати запуску нової траси Random

Також на рисунку 3.4 представлено CDF для часу відгуку від диска WD40EFAX і траси Random з розривом. Максимальна кількість запитів на ввід/вивід, які можуть перебувати в черзі на диску, обмежена 1 (Q-1), 2 (Q-2) і 3 (Q-3).

Показники помилок, отримані для 1, 2 та 3 максимальних запитів в черзі, становлять 0,53 мс, 8,57 мс та 12,29 мс, відповідно крім того при 2 і 3 максимальних запитах у черзі показники значно зростають. Це відбувається тому, що під час відтворення накопичувача деякі специфічні LBN або зміщення з траси Random у певні моменти є недійсними. Під "недійсними" розуміється,

що для конкретного накопичувача агресивні рухи голівки або пошуки від певних LBN призводять до помилок. Цей факт має побічні ефекти: Оскільки деякі запити вимірюються неправильно, наступні запити можуть залишатися в черзі довше, коли неправильно виміряні запити тривають довше, ніж повинні. У тестах для випадкової траси це призводить до зсуву праворуч графіка CDF для 2 і 3 підходів з максимальною кількістю заявок у черзі.

Як видно, для траси Mixed різниця між результатами, отриманими з використанням play і dxreplay, не така помітна. Це пов'язано з двома основними причинами:

Оскільки були активовані звіти про читання і негайний запис, частина запитів обслуговувалася безпосередньо з кешу, уникаючи обслуговування з пластин, а отже, рухів або пошуків головки. Крім того, змішаний слід не такий випадковий і деякі агресивні рухи.

Однак, для обох трас отримані недоліки є досить інформативним, коли максимальна кількість запитів в черзі дорівнює 1. В обох випадках, коли деякі запити вимірюються неправильно, це не впливає на наступні запити, оскільки вони починаються після завершення попередніх запитів. З огляду на отримані результати, коли максимальна кількість заявок в черзі дорівнює 1, можна сказати, що некоректно виміряних заявок практично немає.

Проаналізувавши кожне конкретне робоче навантаження та його наслідки з використанням і без використання дискових буферів, результати були порівняні з симулятором DiskSim. Також було отримано параметри моделі DiskSim для реального диска, запустивши DIXtrac на диску WD40EFAX було відтворене кожне з 5 описаних раніше робочих навантажень у DiskSim і показано їхні CDF-файли разом з запропонованими моделями та реальним диском. Також порівнювались показники дефіциту для кожної траси та обох моделей з реальним диском. Як і в попередньому аналізі, для отримання достовірних результатів спочатку деактивовувались буфери диска, а потім знову активовувались.

Окрім того було порівняно час, необхідний для виконання обох моделей. Спочатку деактивовувались всі процеси вводу/виводу та статистики в обох симуляторах і запускались на виконання для кожної з раніше описаних трас.

Також проведено порівняння прискорення, щоб побачити, наскільки запропонована модель швидша за DiskSim. Також у даному випадку для кожної траси робиться два порівняння (рисунок 3.4) з вимкненими буферами диска і з увімкненими.

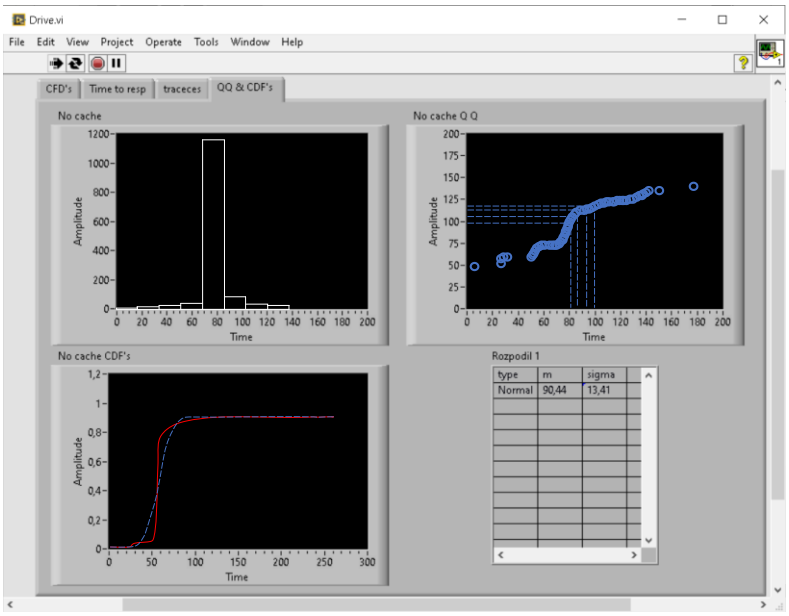


Рисунок 3.4: Гістограма, графіки Q-Q та CDF для часу відгуку диска WD40EFAX та робочого навантаження S3D.

У таблиці на цьому ж рисунку наведено параметри модельованого розподілу. Кешування на диску не активоване

На рисунку 3.4 також показано гістограму, графіки Q-Q та CDF для робочого навантаження S3D. Використання дискового кешу було вимкнено. Було виявлено лише один розподіл, параметри якого показано на рисунку. У цьому робочому навантаженні, хоча доступ до даних є послідовним, розмір кожного запиту становить 2048 блоків. Це призводить до того, що час відгуку довгий. Крім того, оскільки доступ до системи є пакетним, а більшість запитів ставляться в чергу, час відгуку є ще більшим. Отже, на цій трасі час відгуку

переважає над часом очікування, а його значення зосереджені в середньому значенні змодельованого розподілу.

Траса також має кілька коротких часів відгуку, що є наслідком невеликої кількості періодів простою траси. Періоди простою дозволяють уникнути перевантажених звернень, при яких більшість часу запит повинен чекати на обслуговування, поки не закінчаться попередні. Іншою причиною короткого і водночас найдовшого часу відгуку є вплив алгоритму планування диска, який перевпорядковує запити, і ті, що надійшли пізніше (найкоротші), можуть бути обслужені раніше, ніж ті, що надійшли раніше (найдовші). Всі часи відгуку згенеровані з одного розподілу. Як графіки Q-Q, так і графіки CDF показують добру відповідність моделі, а відносна середня похибка становить 8%.

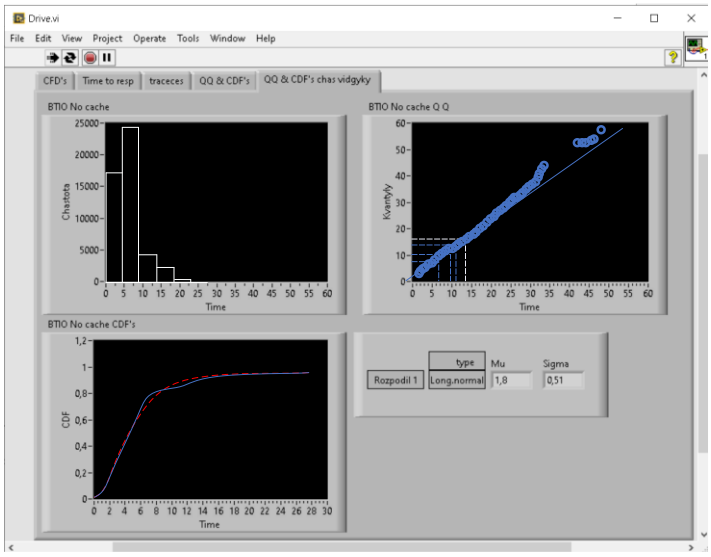


Рисунок 3.5 - Гістограма, графіки Q-Q та CDF для часу відгуку диска WD40EFAX та робочого навантаження BTIO.

У таблиці на рисунку наведено параметри модельованого розподілу. Кешування на диску не активоване

Середній розмір запиту становить 128 блоків, однак, траса не є дуже перевантаженою, тому час відгуку залежить лише від розміру запиту. Довший час відгуку пояснюється невеликою кількістю запитів у трасі або запитами, що надходять після певних періодів простою. Як було сказано раніше, коли траса дуже перевантажена, деякі запити повинні чекати на завершення попередніх, що

збільшує час їхньої відповіді. Для деяких специфічних дисків, після періоду простою, наступний запит, який потрібно обслужити, займає більше часу, ніж зазвичай. Як графіки Q-Q, так і графіки CDF показують відповідність моделі, а відносна середня похибка становить 8,3%. Тут розбіжність у CDF пояснюється помилкою при підборі розподілу.

3.2 Енергозберігаюча архітектура

Для того, щоб оптимізувати кількість збереженої енергії, коли диск перебуває у стані очікування, було запропоновано декілька алгоритмів розкручування [87]. Ці алгоритми допомагають вирішити, коли настає момент переходу зі стану зупинки в стан очікування, щоб диск залишався в стані очікування якомога довше і заощаджував енергію. Системи навчаються на основі попередніх прийнятих рішень і того, як ці рішення спрацювали. Періоди згортання або простою перериваються, коли на диск надходять запити на введення/виведення, читання або запис. Коли алгоритми згортання працюють правильно, вони можуть передбачити, коли на диск надходять запити на введення/виведення, і діють відповідно до них.

Для того, щоб подовжити час очікування в HPC-додатках, пропонується використовувати для дискових накопичувачів підтримуючі SSD-накопичувачі. Таким чином, запити вводу/виводу можуть бути перенаправлені на SSD, досягти як найдовший час очікування. Крім того, вони можуть допомогти заощадити електроенергію, коли час простою недостатньо великий. Твердотільні накопичувачі легкі, безшумні і споживають менше енергії, ніж дискові накопичувачі, оскільки не мають механічних частин. Вони мають лише два стани живлення: активний і режим очікування. SSD знаходиться в активному стані, коли він читає або записує дані. Коли SSD нічого не робить, він перебуває в режимі очікування. Таким чином, при перенаправленні запитів вводу/виводу на підтримуючі SSD диски можуть довше перебувати в стані очікування, заощаджуючи таким чином енергію.

Економія енергії за рахунок обертання дисків вниз тягне за собою компроміс. Щоразу, коли диск обертається вниз і вгору, головки та двигун шпинделя зношуються. Тому виробники вказують максимальну кількість циклів запуску/зупинки, яку диск може витримати без помилок і необхідності заміни. Для накопичувачів для настільних комп'ютерів цей показник становить близько 50 000 циклів запуску/зупинки, а для ноутбуків - близько 300 000.

Крім того, перенаправлення більшості запитів вводу/виводу на підтримуючі SSD означає запис на них великої кількості даних, що також є компромісом. SSD можна записувати обмежену кількість разів. Щоб перезаписати певний блок, його потрібно спочатку стерти і це теж вносить додаткову складність вибору оптимального функціонування. Коли досягається максимальна кількість стирань, блоки SSD стають недоступними для стирання.

Отже, щоб заощадити енергію, зменшуючи швидкість обертання дисків і використовуючи SSD, важливо враховувати обмежену кількість циклів запуску/зупинки в дискових накопичувачах, а також обмежену кількість записів на SSD.

Тому вибір архітектури з урахуванням енергозбереження для великих паралельних обчислювальних середовищ, таких як кластери або суперкомп'ютери першочерговою метою окрім створення «чорної скриньки накопичувача» є створення енергоефективної архітектури на основі SSD, яка може збільшити час простою в HPC-додатках.

На рисунку 3.6 показано загальну архітектуру вводу/виводу для паралельних середовищ. У першому наближенні обчислювальні вузли розміщуються близько до локальної системи зберігання даних. Сучасна тенденція спрямована на відмову від локальних дисків з метою зменшення простору та економії енергії. У таких рішеннях обчислювальні вузли пересилають запити на доступ до файлів до вузлів вводу/виводу, які агрегують і пересилають дані до підсистеми постійного зберігання [88].

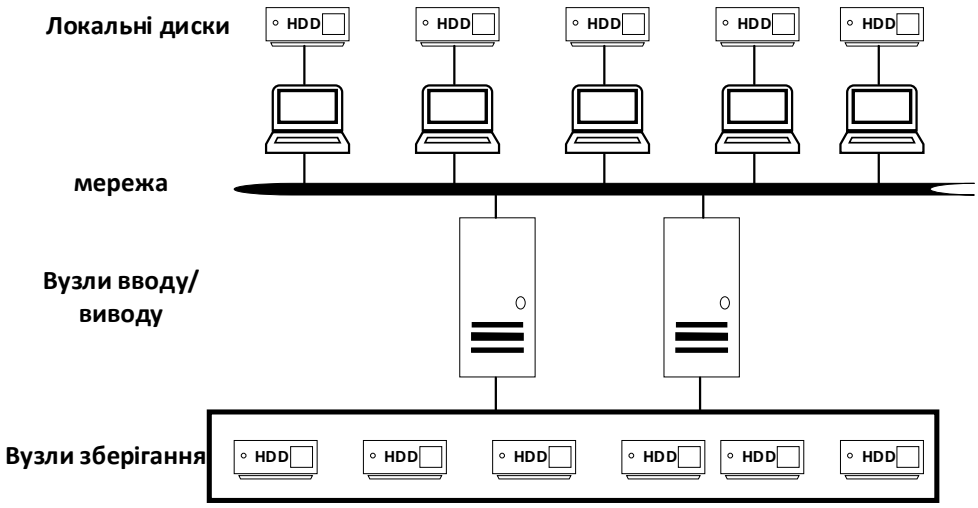
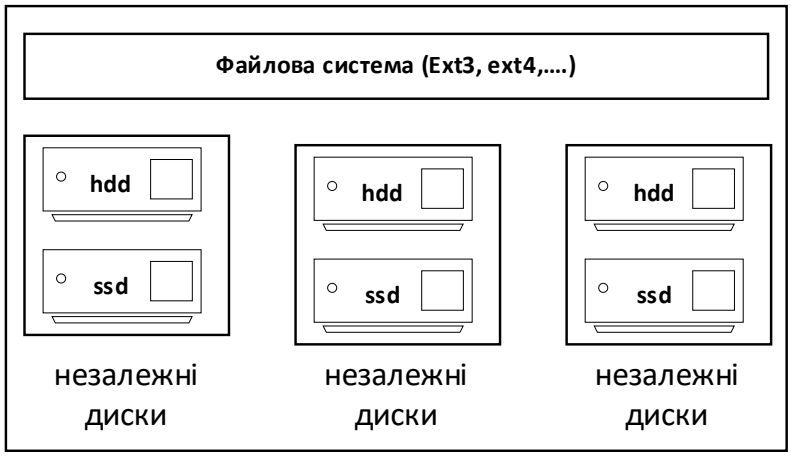
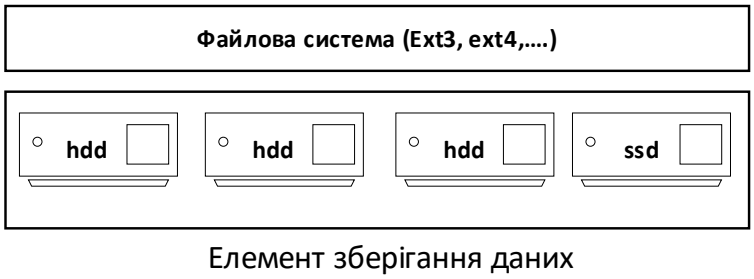


Рисунок 3.6 - Загальна архітектура

Вузли зберігання можуть використовувати переваги SSD-пристроїв наступним чином: незалежні диски можуть мати асоційований SSD-пристрій (як показано на рисунку 3.7.а), або масив незалежних дисків (як показано на рисунку 3.7. б). В обох випадках набір HDD і SSD створює елемент зберігання (ЕЗ).



б) масив незалежних дисків



Елемент зберігання даних

Рисунок 3.7 - Загальна архітектура розширена

Додатки отримують доступ до ЕЗ, як до одного фізичного диска. На рисунку показано, як апаратні компоненти відображаються в реальних підсистемах вводу/виводу. Обчислювальні вузли з'єднані з вузлами вводу/виводу через мережу швидкого з'єднання. Вузли вводу/виводу отримують доступ до системи зберігання від імені обчислювальних вузлів. Сучасна тенденція спрямована на відмову від локальних сховищ для зменшення енергоспоживання обчислювальних вузлів. Вузли зберігання даних забезпечують два підходи: Архітектури на основі незалежних дисків (a), де кожен пристрій HDD має відповідний пристрій SSD, та архітектури на основі RAID-масивів (b), де один SSD працює з масивом незалежних дисків. Всі наступні рішення будуть стосуватись архітектури на основі незалежних дисків.

Тому пропонується архітектура системи зберігання даних з підтримкою енергоефективності на основі твердотільних накопичувачів (SSD-PASS), зображену на рисунку 3.8

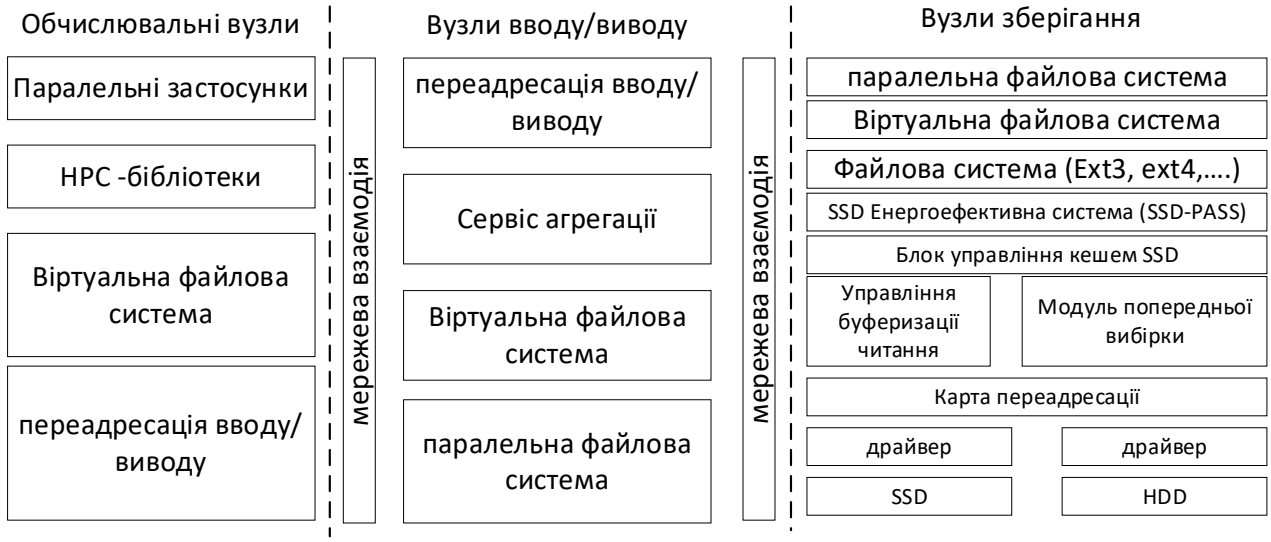


Рисунок 3.8 - Пропонована енергоефективна архітектура

Вона має три рівні: обчислювальні вузли, вузли вводу/виводу та вузли зберігання. Обчислювальні вузли виконують паралельні програми. Додатки отримують доступ до вводу/виводу через бібліотеки, які викликають інтерфейси віртуальних файлових систем, а ті, в свою чергу, викликають інтерфейси

паралельних файлових систем для доступу до вузлів вводу/виводу через мережу. SSD-PASS розгортається на вузлах зберігання, де дані остаточно зберігаються на HDD дисках. SSD-PASS складається з трьох основних модулів: Буферизація запису, попередня вибірка та модулі індексної карти. Також SSD-PASS базується на гібридній архітектурі, в якій кожен вузол вводу-виводу складається з твердотільного накопичувача і звичайного HDD, як показано на рисунку 3.8. Основний акцент ствиться на використання SSD-пристрою, як блочного кешу для певної файлової системи на вузлах вводу/виводу. Операції вводу/виводу є прозорими, як для SSD-пристроїв, так і для паралельної файлової системи.

3.2.1 Доступ до даних

Пропоноване рішення використовує карту переадресації, яка дозволяє розподіляти блоки на різних пристроях, відображаючи адреси логічних блоків (LBA) SSD і диска. Кожен запит на читання і запис надсилається на відповідний пристрій після відображення карти переадресації, використовуючи фактичне фізичне розташування. У системі забезпечується узгодженість даних, оскільки не допускається наявність більше однієї копії файлового блоку на SSD-пристрої. Даний підхід базується на моделі узгодженості з однією копією. Незалежно від того, чи знаходяться дані на одному пристрої, чи на обох, необхідно мати найточнішу інформацію при перерозподілі адрес блоків.

Додатки отримують доступ до архітектури на основі SSD через інтерфейс POSIX, поширений в HPC-додатках [89] і паралельних бібліотеках, таких як MPI. Крім того, пропонована архітектура і система зберігання використовує переваги віртуальної файлової системи, яка віртуалізує операції введення/виведення файлів. Таким чином, для використання нашої архітектури не потрібно модифікувати прикладну та файлову системи.

3.2.2 Політика буферизації запису з урахуванням енергозбереження

Метою традиційної технології буферизації запису є підвищення продуктивності доступу до файлів за рахунок збереження запитів до файлових

даних у буфері пам'яті. Обслуговування запитів з буфера пам'яті відбувається швидше, ніж безпосередньо з дискових накопичувачів.

В даній роботі пропонується енергозберігаюча технологія буферизації з урахуванням енергоспоживання, яка фокусується на збереженні незаписаних блоків у SSD-пристроях до тих пір, поки на них є достатньо місця. Таким чином, періоди, коли диски можуть перебувати в режимі очікування, залежать від розмірів SSD. Чим більший розмір SSD, тим довші періоди простою.

Буфери пам'яті мають дві основні проблеми, коли вони використовуються для енергоефективних цілей. По-перше, вони нестабільні, а це означає, що при відключенні електроенергії кешовані дані будуть втрачені. По-друге, у багатьох операційних системах існують демони, які стежать за тим, щоб кешовані дані час від часу записувалися на диск. Це призводить до того, що тривалі інтервали простою можуть бути розбиті на недостатньо довгі періоди, щоб розкрутити диски і заощадити енергію. Крім того, їхній розмір зазвичай не перевищує кількох гігабайт. Пристрої SSD є енергонезалежними По-друге, вони можуть скористатися перевагами меншої потреби в обслуговуванні, що пов'язано з не надмірним зміщенням головок у дискових пристроях. Як було сказано раніше, SSD не мають механічних частин, а також не мають зсувів головок.

Політика буферизації запису застосовується на SSD і складається з наступних кроків (як показано на Рисунку 3.9):

1) Перенаправлення записів на SSD до тих пір, поки на них є достатньо місця;

2) розкручування дисків;

3) перезапису вмісту SSD назад на диски, коли вони заповнюються.

Перенаправлення записів на SSD робиться у вигляді журналу, після чого знаходиться вміст поточного запиту після попередніх. Таким чином, використовується перевага послідовного запису на SSD. Крім того, якщо запити є певною мірою послідовними, то при очищенні вмісту SSD легше впорядкувати цей вміст.

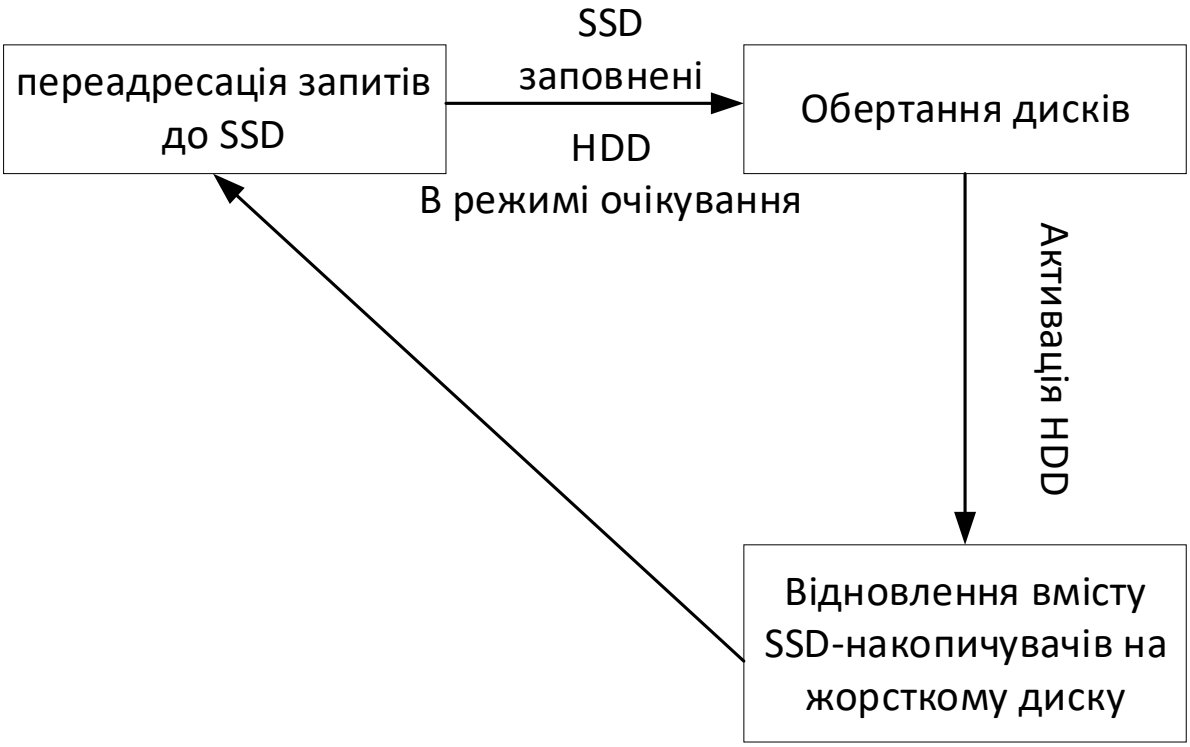


Рисунок 3.9 – Політика буферизації

Після деякого часу перебування в режимі очікування диски повинні розкрутитися, щоб бути готовими до прийому записаних даних з твердотільних накопичувачів. Як було сказано вище, розкручування дисків займає певний час (порядку декількох секунд) і споживає певну кількість енергії. Щоб програми не витрачали час на розкручування дисків, їх можна розкрутити заздалегідь. У цьому випадку розкручування дисків перед заповненням SSD повинно тривати стільки ж секунд, скільки потрібно для розкручування дисків. Таким чином, коли твердотільні накопичувачі будуть заповнені, дискові накопичувачі будуть готові прийняти записаний на них вміст. Попереднє розкручування дисків може споживати більше енергії, але це зменшує час запуску програм.

Відновлення вмісту SSD-накопичувачів, коли вони заповнені, потребує певного часу та втрат енергії. Першим кроком є зчитування записаного вмісту з твердотільних накопичувачів. Потім дані переупорядковуються, щоб скоротити час на перезапис вмісту на диски. Після того, як дані впорядковані, вміст записується назад на диски. Час, необхідний для очищення, в основному

залежить від розміру твердотільних накопичувачів. Чим більші SSD, тим більше часу займе очищення.

3.2.3 Політика попередньої вибірки з урахуванням енергозбереження

Як було сказано в попередньому розділі, деякі HPC-додатки демонструють повторювану поведінку, коли раніше записані дані зчитуються знову. Ця особливість робить оптимальною описану вище політику буферизації запису, оскільки дискові накопичувачі розкручуються лише тоді, коли SSD заповнені, а не тому, що деякі зчитані дані не можуть бути знайдені на SSD. Однак, інші програми надають доступ на читання до даних, які не можуть бути знайдені на SSD. Це пов'язано з тим, що ці дані не були раніше записані або були перезаписані іншими даними. Щоб уникнути такої ситуації, на SSD можна зарезервувати певну область, в якій зберігатимуться дані, до яких передбачається доступ у майбутньому. Це допомагає усунути непотрібні спін-апи. Для того, щоб максимально ефективно використовувати цю область з метою енергозбереження, в цій тезі пропонується декілька політик попередньої вибірки.

Метою звичайної техніки попередньої вибірки є покращення доступу до файлу за допомогою попереднього запиту даних до файлу. Однак в даній роботі акцент був на енергозберігаючу попередню вибірку, яка фокусується на зчитуванні заздалегідь достатньої кількості блоків in або der, щоб уникнути переривання довгих інтервалів простою. Якщо робоче навантаження є достатньо послідовним, читання послідовних блоків на SSD-пристрій за допомогою однієї операції вводу/виводу може задовольнити майже всі запити на читання під час тривалого простою. Крім того, вона може скористатися перевагами коротших інтервалів між запитами на обслуговування, які виникають через неекстремальні зміщення головок на дискових накопичувачах.

Політика попередньої вибірки виконується на SSD і складається з: прийняття рішення про те, які блоки попередньо вибирати (передбачення), моніторингу попередньої вибірки блоків і переміщення попередньо вибраних

блоків з кешу на основі SSD до додатків. Для деяких політик модуль попередньої вибірки налаштовує параметри в режимі реального часу, наприклад, розміри вікон, щоб отримати найкращу продуктивність для додатків, що інтенсивно використовують дані.

3.3 Тестування моделей на основі кількох реальних трас

Щоб продемонструвати працездатність моделі, було проведено тестування декількох ненавчених трас, використовуючи попередньо побудовану модель чорного ящика. Під ненавченими трасами розуміються траси, які раніше не використовувалися при побудові моделей. Було вибрано декілька трас блочного вводу/виводу з репозиторію SNIA IOTTA [90]:

- WebResearch представляє веб-управління декількома проектами з використанням веб-сервера Apache.

- Online представляє систему управління курсами кафедри, що використовують Moodle.

- WebUsers - веб-сервер, на якому розміщені веб-сайти викладачів, співробітників та аспірантів.

- WebMail представляє веб-інтерфейс до веб-сервера.

На рисунках 3.10, 3.11, 3.12 і 3.13, які представлено нижче показано Q-Q і CDF результати виконання чотирьох раніше описаних трас на генераторі змінних (bbm) і диску WD40EFAX.

Було побудовано криві Q-Q і CDF для реального диска і для моделі, а також було використано похибку [91], як метрику для перевірки пропонованої моделі в порівнянні з реальним диском. Показники похибки представлені у відносному вираженні (у відсотках від середнього часу відгуку).

У цьому випадку модель чорного ящика (bbm) базується на п'яти реальних трасах. Як було сказано раніше, для прогнозування часу відгуку за допомогою моделі чорної скриньки (bbm) обирається одна із змодельованих трас. Модель обирає ту, яка має схожі характеристики розміру, часу очікування в черзі та послідовності.

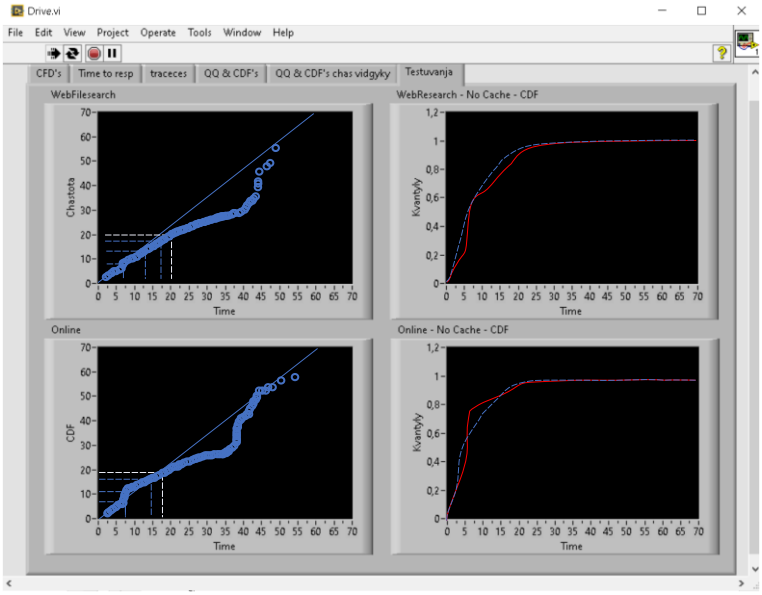


Рисунок 3.10 - Графіки CDF та QQ для часу відгуку, отримані за допомогою моделі чорної скриньки (bbm).

На рисунках 3.14 та 3.15 показано візуальне порівняння Q-Q та CDF для чотирьох трас, виконаних за допомогою деактивації буферної кеш-пам'яті на диску. Недоліки цих розподілів показано в таблиці 3.1. Для чотирьох трас прогнози зроблено на основі моделі, побудованої на рисунку 3.16.

В даному випадку виконувались ненавчені траси. Кешування на диску не активовано.

Як WebResearch, так і Online (див.рис. 3.10) демонструють найгірші результати з чотирьох трас. Помилки пов'язані з основною причиною: Час очікування в черзі. Для траси

Як для WebResearch, так і для Online час очікування в черзі зустрічається частіше, ніж для інших, і реальний час відповіді виявляється довшим, ніж той, що згенерований. Це відбувається тому, що якщо поточний запит був довгим, його тривалість не впливає на наступні запити. Це пояснюється тим, що час відгуку генерується незалежно, і якщо поточний час відгуку був довгим, то наступний запит не обов'язково повинен бути таким же довгим. У конкретному випадку трасування Online багато ефектів черги призводять до того, що час відгуку має бути меншим, ніж значення, які згенеровані.

Крім того, як для WebResearch, так і для Online, час відгуку в діапазоні, що належить до частини 1 Розподілу 1, є меншим, ніж реальний час відгуку для того ж діапазону. Це пов'язано з тим, що коли поточний запит ставиться в чергу, і він має чекати менше 4 секунд, час його відповіді формується з частини 1 Дистрибутиву 1, і знову ж таки не залежить від попередніх запитів. Це означає, що поточний час відповіді може бути коротшим, ніж реальний час запиту, як це буває.

На рисунках 3.11 та 3.12 показано візуальне порівняння Q-Q та CDF для чотирьох трас, запущених на диску з буферною кеш-пам'яттю. Недоліки цих розподілів також показано в таблиці 3.1. Для чотирьох трас зроблено прогнози на основі моделі, побудованої на рисунку 3.12. Тут недоліки дещо вищі. Це пов'язано з використанням кешу, який дає менший час відгуку, і середнє значення також є меншим. Обчислення відносної похибки передбачає ділення на це середнє значення, і неважливо, наскільки короткою є різниця між CDF, коли різниця є занадто довгою, це впливає на відносну похибку. У цьому випадку похибки також зумовлені тією ж причиною -Час очікування в черзі. Вплив їх більш помітний після обслуговування запиту, коли диск простоює більше секунди.

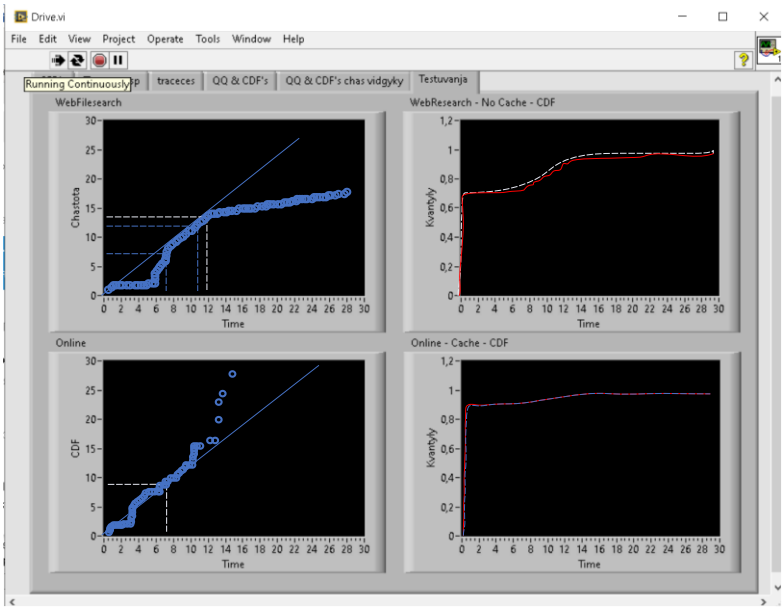


Рисунок 3.11 - Розбіжності між реальними та змодельованими значеннями

Як видно з таблиці 3.1, найнижчий показник при активному кешуванні отримано для веб-пошти (25%). Це пояснюється тим, що діапазон часу відгуку охоплює більші значення, а також тим, що великі розбіжності між реальними та змодельованими значеннями спостерігаються не дуже часто. Наступний недолік отримано для WebResearch та Online (31%). У WebResearch діапазон часу відгуку охоплює найбільші значення, тому різниця між реальним та змодельованим часом відгуку не надто впливає на розрахунок балу. В Online та WebUsers діапазон часу відповіді не охоплює високих значень, але невеликі відмінності між реальним та змодельованим часом відповіді впливають на розрахунок де-меріті, особливо для WebUsers (39%).

Таблиця 3.1 - Недоліки у відносному вираженні для запропонованого підходу на основі моделі чорної скриньки (bbm). Виконуються ненавчені траси.

	Cache	No Cache
WebResearch	31%	24%
Online	31%	25%
WebUsers	32%	21%
Webmail	25%	21%

3.3.1 Тестування моделей на основі синтетичної траси

Як було сказано раніше, для побудови моделі на основі синтетичної траси спочатку отримуються експериментальні дані з реального диска. При побудові моделі не враховується час очікування в черзі. Отже, при отриманні експериментальних даних є обмеження на максимальну кількість запитів при вводі/виводі, які можуть стояти в черзі на диску.

Час очікування в черзі не враховується, оскільки в запропонованій моделі час очікування моделюється на льоту, що робить її більш загальною та універсальною.

Для синтетичної траси, серед різних доступних варіантів, було обрано робоче навантаження SPC Bench mark [92], визначене Storage Performance Council. Як зазначено в [93], траси, згенеровані бенчмарком, імітують реальні

умови, характерні для типових додатків, таких як OLTP-системи, системи баз даних і додатки поштових серверів.

При побудові моделі будується гістограма для даних, отриманих за допомогою play (програми для вимірювання часу обслуговування). Представляючи гістограму, визначаються можливі розподіли, до яких можна підігнати модель. Після цього «підганяються» знайдені раніше розподіли до теоретичних.

Потім знаходяться причини, і на основі цих причин будується модель, тобто знову відтворюється синтетична траса на щойно побудованій моделі і виводиться Q-Q графік, щоб оцінити відповідність моделі експериментальним даним, і CDF, щоб побачити, наскільки добре вони накладаються. При тестуванні моделі відтворювались кілька реальних.

Для реальних трас було обрано дві з раніше описаних трас. Було вибрано ті, які були більш схожі за характеристиками на навантаження SPC-1. Зокрема, обирались ті, що були найбільш схожі за діапазоном розмірів запитів до робочих навантажень SPC-1. Цими обраними трасами були Financial та Cello. Як вже було сказано раніше, Financial - це ядро вводу/виводу OLTP-додатку, зібране у фінансовій організації. З іншого боку, Cello - це спільний обчислювальний/поштовий сервер від HP Labs. Щоб побачити ефект онлайн-обчислення часу запиту, також використовувалась інша траса - WebSearch. Це сервер пошукової системи, який виконує близько 4 мільйонів запитів на читання на 6 дисках лише за 4 години.

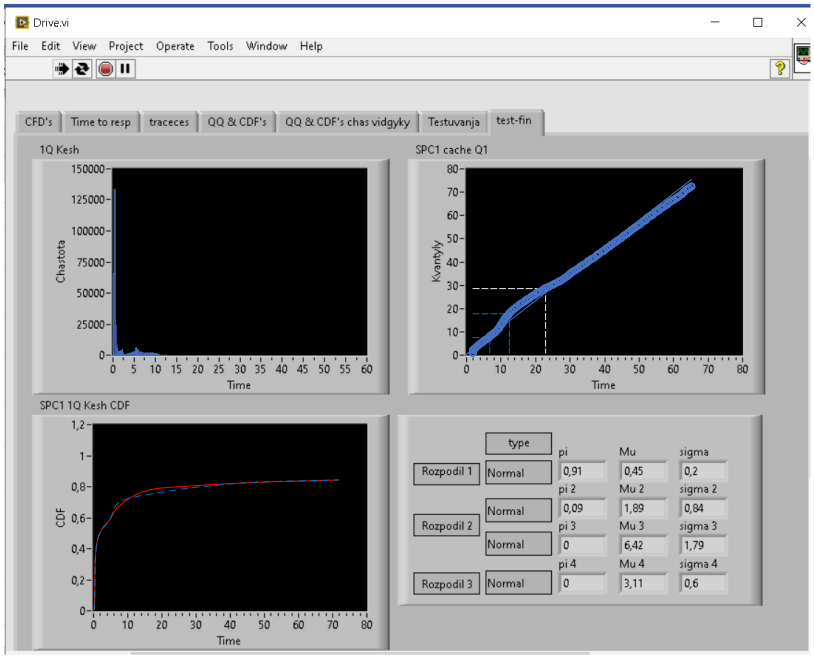


Рисунок 3.12 - Гістограма, графіки Q-Q та CDF для часу відгуку диска WD40EFAX та робочого навантаження SPC-1.

Для простоти роботи, деактивація буферів не проводилась. Модель будувалась і проводилось прогноування на її основі, використовуючи за замовчуванням випереджувальне читання і негайне звітування. На рисунку 3.12 показано гістограму, графіки Q-Q і CDF для робочого навантаження SPC-1 і диска WD40EFAX.

У таблиці на рисунку наведено параметри змодельованих розподілів. Кешування на диску активне. Час очікування в черзі в моделі не враховується.

Як вже говорилось раніше, було активовано використання кешу диска. Було виділено три дистрибутиви, параметри яких наведено в таблиці на рисунку. Часи відгуку дистрибутива 1 - частина 1 моделюють негайний відгук від звернень до кешу запису. Вони також моделюють звертання до кешу читання, які були попередньо передбачені. Інші два дистрибутиви моделюють доступ до пластин, які не можуть обслуговуватися дисковим кешем. Розподіл 2 моделює час відгуку при послідовному доступі, а розподіл 3 моделює час відгуку при непослідовному доступі. Розподіл 2 також моделює час відгуку для запитів на запис, розмір яких перевищує 64 блоки. Час очікування не враховується у визначених розподілах. Як графіки Q-Q, так і графіки CDF показують задовільну відповідність моделі, а відносна середня похибка становить 25.5%.

Висновки до розділу 3:

У цьому розділі представлено експериментальну оцінку описаної раніше моделі чорної скриньки для дискового накопичувача, так і запропонованих енергозберігаючих методів.

Експерименти демонструють точність моделі чорної скриньки для навантажень з подібними характеристиками розміру, часу очікування в черзі та послідовності, ніж одна з попередньо навчених трас, використаних при побудові моделі.

На основі роботи чорної скриньки накопичувача бу запропоновано енергозберігаючу архітектуру для HPC додатків, описано енергоощадні політики буферизації запису, вибірки та доступу до даних.

Також продемонстровано, що моделі на основі синтетичних трас можуть бути менш точними, але більш загальними та універсальними. Результати показано для робочих навантажень з подібними характеристиками розміру та послідовності, як і у навченої синтетичної траси. Також політика буферизації запису зменшує енергоспоживання,

Було оцінено різні розміри підтримуваних SSD-накопичувачів, і експерименти показали, що архітектури, які використовують SSD-накопичувачі менші за 100 ГБ, навіть не амортизують початкові бюджети. Це пов'язано з все ще високими цінами на SSD.

Експерименти для синтетичних і реалістичних робочих навантажень демонструють, що політика послідовного випередження добре працює для послідовних додатків.

Результати показують, що методи моделювання "чорної скриньки" можуть бути використані для точного енергетичного моделювання.

ВИСНОВКИ

При виконанні даної роботи був проведеней аналіз досліджень в області моделювання та оптимізації роботи накопичувачів, було досліджено принципи функціонування та конструкційні особливості твердотільних та жорстких накопичувачів. Окрім того було проаналізовано методи та засоби емулювання і моделювання накопичувачів. Також був проведений аналіз можливості застосування аналітичних моделей, як точної але більш складної альтернативи чорній скринці. Крім цього був проведений аналіз ключового напрямку, а саме: енергозберігаючі застосування для накопичувачів, що дозволяють зменшити та економити енергію споживання, за рахунок методів попередньої обробки і кешування, міграції даних між накопичувачами.

На основі проведених досліджень було запропоновано метод побудови моделі чорної скриньки для жорстких дисків, він включає в себе вимірювання часу обслуговування як ключового фактора для побудови чорної скриньки накопичувача, що мало залежна від марки та типу інтерфейсу які будуть використовуватись. Також запропоновано підходи побудови таких моделей, що базується на декількох реальних трасах і одній або більше синтетичних трасах. З таким підходом забезпечується більш точніший і загальний альтернативний підхід для моделювання часу обслуговування для накопичувача, незалежно від інтерфейсу, тому, що, реалізація вимірювання часу обслуговування використовує стандартні виклики POSIX. Крім того, модель чорної скриньки накопичувача не потребує багато додаткових характеристик від виробників та тестувальників.

Також в роботі було реалізовано модель чорної скриньки для дискового накопичувача та проведено експериментальну оцінку описаної раніше моделі, і енергозберігаючих методів. Проведені експерименти продемонстрували точність. Базуючись на результатах чорної скриньки накопичувача, запропоновано енергозберігаючу архітектуру для HPC додатків.

Результати екперементів та емуляції показали, що HPC архітектури, які використовують SSD-накопичувачі менші за 100 ГБ, не впливатимуть на зменшення енергоспоживання. Результати експериментів реальних робочих навантажень і синтетичних показують, що управління послідовним випередженням найкраще підходить до послідовних додатків. Також результати показують, що метод моделювання "чорної скриньки" може бути використаний для точного аналізу енергоспоживання.

Подальшим напрямком наукових досліджень можна віднести розробку системи управління накопичувачами які безпосередньо працюють на серверах дистанційного навчання, так як там є проблеми з нерівномірною роботою та невиправданим споживанням енергії.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Yadgar, Gala, et al. SSD-based workload characteristics and their performance implications. ACM Transactions on Storage (TOS), 2021, 17.1: 1-26.
2. Nand512w3a Datasheet, https://www.alldatasheet.com/datasheet-pdf/pdf/94124/STMICROELECTRONICS/NAND512W3A.html
3. Guo, Jiayang. Novel Methods for Improving Performance and Reliability of Flash-Based Solid State Storage System. 2018. PhD Thesis. University of Cincinnati
4. Kumar, Shailesh; DUBEY, Kumkum; SINGH, P. K. A survey on flash translation layer for NAND flash memory. Indian J. Sci. Technol, 2018, 11: 1-7.
5. Luo, Yuhan; LIN, Mingwei. Flash translation layer: a review and bibliometric analysis. International Journal of Intelligent Computing and Cybernetics, 2021, 14.3: 480-508.
6. Alsalibi, Ahmed Izzat, et al. A survey of techniques for architecting SLC/MLC/TLC hybrid Flash memory–based SSDs. Concurrency and Computation: Practice and Experience, 2018, 30.13: e4420.
7. Трушаков, Д. В.; НУЖНИЙ, В. В. Застосування нового типу пам’яті 3D V-NAND в SSD дисках та розгляд їх переваг. НАУКОВІ ЗАПИСКИ, 2019, 8.
8. Fukuchi, Mamoru, et al. 20% system-performance gain of 3D charge-trap TLC NAND flash over 2D floating-gate MLC NAND flash for SCM/NAND flash hybrid SSD. In: 2018 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2018. p. 1-5.
9. Булатецький, В. В.; БУЛАТЕЦЬКА, Л. В.; СТУПІНЬ, А. П. ТЕХНОЛОГІЇ ТВЕРДОТІЛЬНИХ НАКОПИЧУВАЧІВ. Прикладні проблеми комп’ютерних наук, безпеки та математики, 2023, 1: 20-27.
10. Neumann, Thomas; FREITAG, Michael J. Umbra: A Disk-Based System with In-Memory Performance. In: CIDR. 2020. p. 29.
11. NIU, Junpeng; XU, Jun; XIE, Lihua. Hybrid storage systems: A survey of architectures and algorithms. IEEE Access, 2018, 6: 13385-13406.

12. Узденов, Т. А. Симулятор процесу диспетчеризації задач в GRID-системах з невідчужуваними ресурсами. Elektronnoe Modelirovanie, 2021, 43.1.
13. Dartois, Jean-Emile, et al. Investigating machine learning algorithms for modeling ssd i/o performance for container-based virtualization. IEEE transactions on cloud computing, 2019, 9.3: 1103-1116.
14. Zou, Lida, et al. Penalty Cost Minimization for Multi-tenant Query Deadline Employing Cache Optimization and Log Based Dispatching. In: 2016 IEEE Trustcom/BigDataSE/ISPA. IEEE, 2016. p. 1248-1255.
15. Пунда, Сергій Юрійович. Дослідження впливу методів економії ємності на продуктивність системи зберігання даних. 2021. Master's Thesis. КПІ ім. Ігоря Сікорського.
16. Kim, Joonsung, et al. SSDcheck: Timely and accurate prediction of irregular behaviors in black-box SSDs. In: 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2018. p. 455-468.
17. Akgun, Ibrahim Umit. Using Machine Learning to Improve Operating Systems' I/O Subsystems. 2022. PhD Thesis. State University of New York at Stony Brook.
18. Kim, Joonsung, et al. Performance modeling and practical use cases for black-box SSDs. ACM Transactions on Storage (TOS), 2021, 17.2: 1-38.
19. THOMASIAN, Alexander. Mirrored and hybrid disk arrays: Organization, scheduling, reliability, and performance. arXiv preprint arXiv:1801.08873, 2018.
20. kishani, Mostafa; AHMADIAN, Saba; ASADI, Hossein. A modeling framework for reliability of erasure codes in ssd arrays. IEEE Transactions on Computers, 2019, 69.5: 649-665.
21. Von Merzljak, Leonard, et al. What Are You Waiting For? Use Coroutines for Asynchronous I/O to Hide I/O Latencies and Maximize the Read Bandwidth!. In: International Workshop on Accelerating Data Management Systems (ADMS). 2022.
22. Ben-David, Naama, et al. Implicit decomposition for write-efficient connectivity algorithms. In: 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE, 2018. p. 711-722.

23. Fevgas, Athanasios, et al. Indexing in flash storage devices: a survey on challenges, current approaches, and future trends. The VLDB Journal, 2020, 29: 273-311.
24. Chakraborttii, Chandranil; LITZ, Heiner. Learning i/o access patterns to improve prefetching in ssds. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Cham: Springer International Publishing, 2020. p. 427-443.
25. Park, Jonggyu; EOM, Young Ik. Fragpicker: A new defragmentation tool for modern storage devices. In: Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. 2021. p. 280-294.
26. The DiskSim https://www.pdl.cmu.edu/DiskSim/index.shtml,
27. Dartois, Jean-Emile, et al. Investigating machine learning algorithms for modeling ssd i/o performance for container-based virtualization. IEEE transactions on cloud computing, 2019, 9.3: 1103-1116.
28. Zhu, Guangyu; LEE, Sang Jin; SON, Yongseok. An efficient log-structured scheme for disk arrays. In: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing. 2022. p. 1197-1204.
29. Su, Chuan-Jun; HUANG, Shi-Feng. Real-time big data analytics for hard disk drive predictive maintenance. Computers & Electrical Engineering, 2018, 71: 93-101.
30. Thomasian, Alexander. Mirrored and hybrid disk arrays: Organization, scheduling, reliability, and performance. arXiv preprint arXiv:1801.08873, 2018.
31. Wilcox, Peter; LITZ, Heiner. Design for computational storage simulation platform. In: Proceedings of the Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems. 2021. p. 1-8.
32. Long, Darrell. Swift: Using distributed disk striping to provide high I/O data rates. 2023.
33. De Santo, Aniello, et al. Deep Learning for HDD health assessment: An application based on LSTM. IEEE Transactions on Computers, 2020, 71.1: 69-80.

34. Yao, Yingbiao, et al. Hdftl: An on-demand flash translation layer algorithm for hybrid solid state drives. IEEE Transactions on Consumer Electronics, 2021, 67.1: 50-57.
35. Kim, Bryan S.; CHOI, Jongmoo; MIN, Sang Lyul. Design tradeoffs for {SSD} reliability. In: 17th USENIX Conference on File and Storage Technologies (FAST 19). 2019. p. 281-294.
36. Kwak, Jaewook, et al. Cosmos+ OpenSSD: Rapid prototype for flash storage systems. ACM Transactions on Storage (TOS), 2020, 16.3: 1-35.
37. Kong, Lingjun, et al. A gradient heatmap based table structure recognition. In: 2021 13th International Conference on Machine Learning and Computing. 2021. p. 456-463.
38. Randau, Simon, et al. Benchmarking the performance of all-solid-state lithium batteries. Nature Energy, 2020, 5.3: 259-270.
39. Zacharis, Nick Z. Classification and regression trees (CART) for predictive modeling in blended learning. IJ Intelligent Systems and Applications, 2018, 3.1: 9.
40. Dartois, Jean-Emile, et al. Investigating machine learning algorithms for modeling ssd i/o performance for container-based virtualization. IEEE transactions on cloud computing, 2019, 9.3: 1103-1116.
41. Kaur, Kamaljit; KAUR, Kuljit. Failure Prediction, Lead Time Estimation and Health Degree Assessment for Hard Disk Drives Using Voting Based Decision Trees. Computers, Materials & Continua, 2019, 60.3.
42. Borowiec, Benjamin; NOONAN, Terence. Proactive management of a plurality of storage arrays in a multi-array system. U.S. Patent No 10,162,835, 2018.
43. Carrizosa, Emilio; MOLERO-RÍO, Cristina; ROMERO MORALES, Dolores. Mathematical optimization in classification and regression trees. Top, 2021, 29.1: 5-33.
44. Kim, Joonsung, et al. Performance modeling and practical use cases for black-box SSDs. ACM Transactions on Storage (TOS), 2021, 17.2: 1-38.

45. HAO, Mingzhe, et al. {LinnOS}: Predictability on Unpredictable Flash Storage with a Light Neural Network. In: 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). 2020. p. 173-190.
46. Yoshikawa, Takao, et al. Storage system and storage system management method. U.S. Patent No 10,628,083, 2020.
47. Liberatore, Matthew J.; WAGNER, William P. User performance on laptops vs. tablets: an experiment in the field. Behaviour & Information Technology, 2022, 41.13: 2878-2886.
48. Dong, Yong, et al. Lazy scheduling based disk energy optimization method. Tsinghua Science and Technology, 2019, 25.2: 203-216.
49. Giardino, Michael, et al. Low-Overhead Reinforcement Learning-Based Power Management Using 2QoSM. Journal of Low Power Electronics and Applications, 2022, 12.2: 29.
50. Taherikhonakdar, Zahra; FAZLOLLAHTABAR, Hamed. Proposing the method for Operating System energy consumption measurement with energy efficiency approach. Environmental Progress & Sustainable Energy, 2023, e14072.
51. Oleksiak, Ariel, et al. Energy aware ultrascale systems. 2019
52. Bostoen, T., Mullender, S., & Berbers, Y. (2013). Power-reduction techniques for data-center storage systems. ACM Computing Surveys (CSUR), 45(3), 1-38.
53. Lvarez Valera, Hernn H., et al. The architecture of kaligreen v2: A middleware aware of hardware opportunities to save energy. In: 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). IEEE, 2019. p. 79-86.
54. Jain, Nishant, et al. Extending battery life of smartphones by overcoming idle power consumption using ambient light energy harvesting. In: 2018 IEEE International Conference on Industrial Technology (ICIT). IEEE, 2018. p. 978-983.
55. Kang, Mincheol; LEE, Wonyoung; KIM, Soontae. Subpage-aware solid state drive for improving lifetime and performance. IEEE Transactions on Computers, 2018, 67.10: 1492-1505.

56. Song, Minseok. Minimizing power consumption in video servers by the combined use of solid-state disks and multi-speed disks. IEEE Access, 2018, 6: 25737-25746.
57. Liao, Chi-Shun; CHUANG, Hui-Kai. Determinants of innovative green electronics: An experimental study of eco-friendly laptop computers. Technovation, 2022, 113: 102424.
58. Roberts, David A. Method and apparatus for controlling cache line storage in cache memory. U.S. Patent No 11,237,972, 2022.
59. Micheloni, Rino; CRIPPA, Luca; PICCA, M. Hybrid storage systems. In: Inside Solid State Drives (SSDs). Singapore: Springer Singapore, 2018. p. 43-59.
60. Wang, Kefei; CHEN, Feng. Cascade mapping: Optimizing memory efficiency for flash-based key-value caching. In: Proceedings of the ACM Symposium on Cloud Computing. 2018. p. 464-476.
61. Dulong, Rémi, et al. NVCache: A plug-and-play NVMM-based I/O booster for legacy systems. In: 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2021. p. 186-198.
62. Yadgar, Gala, et al. SSD-based workload characteristics and their performance implications. ACM Transactions on Storage (TOS), 2021, 17.1: 1-26.
63. Gao, Congming, et al. Constructing large, durable and fast SSD system via reprogramming 3D TLC flash memory. In: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. 2019. p. 493-505.
64. Xiao, Jiang, et al. Disk failure prediction in data centers via online learning. In: Proceedings of the 47th International Conference on Parallel Processing. 2018. p. 1-10.
65. Akgun, Ibrahim Umit, et al. Improving Storage Systems Using Machine Learning. ACM Transactions on Storage, 2023, 19.1: 1-30.
66. Borba, Eric; TAVARES, Eduardo; MACIEL, Paulo. A modeling approach for estimating performance and energy consumption of storage systems. Journal of Computer and System Sciences, 2022, 128: 86-106.

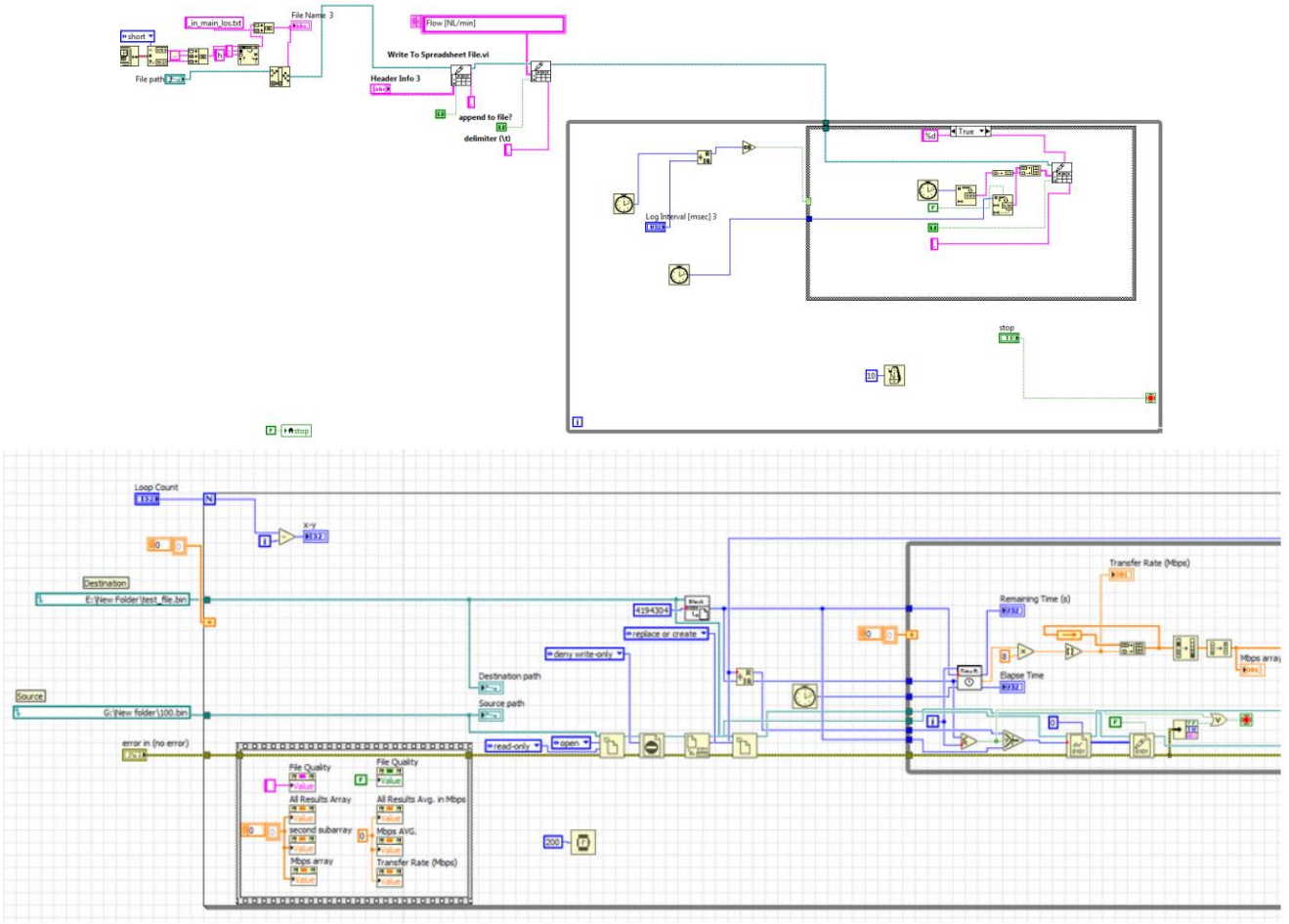
67. Gezelter, Robert. Revisiting Operating System Mass Storage Presumptions Enables Higher Performance and Efficiency. In: 2022 IEEE Long Island Systems, Applications and Technology Conference (LISAT). IEEE, 2022. p. 1-9.
68. Hao, Mingzhe. Fast and Stable Data and Storage Systems in Milli/Micro-Second Era. 2020. PhD Thesis. The University of Chicago.
69. Mohammed, Ali, et al. An approach for realistically simulating the performance of scientific applications on high performance computing systems. Future Generation Computer Systems, 2020, 111: 617-633.
70. Srinivas, A. Vijay; JANAKIRAM, D. Data management in distributed systems: A scalability taxonomy. Scalable Computing: Practice and Experience, 2007, 8.1.
71. Karniavoura, Flora; MAGOUTIS, Kostas. Decision-making approaches for performance QoS in distributed storage systems: A survey. IEEE Transactions on Parallel and Distributed Systems, 2019, 30.8: 1906-1919.
72. Hou, Binbing. On I/O Performance and Cost Efficiency of Cloud Storage: A Client's Perspective. Louisiana State University and Agricultural & Mechanical College, 2019.
73. VM, HITACHI UNIFIED STORAGE. SPC BENCHMARK 1™ FULL DISCLOSURE REPORT HITACHI DATA SYSTEMS CORPORATION. 2014.
74. Dube, Parijat, et al. Performance of large low-associativity caches. ACM SIGMETRICS Performance Evaluation Review, 2010, 37.4: 11-18.,
75. Harris, Charles R., et al. Array programming with NumPy. Nature, 2020, 585.7825: 357-362.
76. Estro, Tyler, et al. Desperately Seeking... Optimal {Multi-Tier} Cache Configurations. In: 12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 20). 2020.
77. The R Project for Statistical Computing, 2011. Available at http://www.r-project.org/
78. Rademacher, Tim, et al. Insights into source/sink controls on wood formation and photosynthesis from a stem chilling experiment in mature red maple. New Phytologist, 2022, 236.4: 1296-1309.

79. Sun, Hao; Yang, Xianqiang; Gao, Huijun. A spatially constrained shifted asymmetric Laplace mixture model for the grayscale image segmentation. Neurocomputing, 2019, 331: 50-57.
80. UMass Trace Repository// umass.edu [Електронний ресурс]. - Режим доступу: https://traces.cs.umass.edu/
81. Brinkmann, André, et al. Ad hoc file systems for high-performance computing. Journal of Computer Science and Technology, 2020, 35: 4-26.
82. OMNeT++ 6.0.2// omnetpp.org [Електронний ресурс]. - Режим доступу: http://www.omnetpp.org/
83. Vigna, Sebastiano. It is high time we let go of the Mersenne Twister. arXiv preprint arXiv:1910.06437, 2019.
84. WD Red NAS Hard Drive // [Електронний ресурс]. - Режим доступу: westerndigital.com https://www.westerndigital.com/en-ap/products/internal-drives/wd-red-sata-hdd?sku=WD20EFAX
85. What Is LabVIEW?/ ni.com [Електронний ресурс]. - Режим доступу: /https://www.ni.com/en/shop/labview.html
86. Wu, Fenggang; LI, Bingzhe; DU, David HC. Fluidsmr: Adaptive management for hybrid smr drives. ACM Transactions on Storage (TOS), 2021, 17.4: 1-30.
87. Dong, Yong, et al. Lazy scheduling based disk energy optimization method. Tsinghua Science and Technology, 2019, 25.2: 203-216., OKONOR, Ogechukwu M. Improving Energy Efficiency in Cloud Computing Data Centres Using Intelligent Mobile Agents. 2021. PhD Thesis. University of Portsmouth
88. Zheng, Huihuo, et al. HDF5 Cache VOL: Efficient and Scalable Parallel I/O through Caching Data on Node-local Storage. In: 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid). IEEE, 2022. p. 61-70.
89. Shirvani, Mirsaeid Hosseini; RAHMANI, Amir Masoud; SAHAFI, Amir. A survey study on virtual machine migration and server consolidation techniques in DVFS-enabled cloud datacenter: taxonomy and challenges. Journal of King Saud University-Computer and Information Sciences, 2020, 32.3: 267-286.

90. I/O Trace Data Files// snia.org [Електронний ресурс]. - Режим доступу: http://iotta.snia.org/traces
91. Jin, Hongyue, et al. Life cycle assessment of emerging technologies on value recovery from hard disk drives. Resources, Conservation and Recycling, 2020, 157: 104781.
92. SPC benchmark specification// storageperformance.org [Електронний ресурс]. - Режим доступу:https://www.storageperformance.org/benchmarks
93. Du, Xiaoming; LI, Cong. SHARC: improving adaptive replacement cache with shadow recency cache management. In: Proceedings of the 22nd International Middleware Conference. 2021. p. 119-131.
94. Загальні рекомендації з підготовки, оформлення, захисту та оцінювання випускних кваліфікаційних робіт здобувачів вищої освіти першого «бакалаврського» і другого «магістерського» рівнів / За ред. доц. М.І. Шинкарика. Тернопіль: ТНЕУ, 2018. 67 с.
95. Комар М.П., Саченко А.О., Васильків Н.М. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за другим (магістерським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2021. 32 с.
96. Іваночко В. Р., Осолінський О.Р. "Метод моделювання "чорної скриньки" для пристроїв зберігання даних". Цифровізація науки та сучасні тренди її розвитку: матеріали V Міжнародної студентської наукової конференції, м. Житомир, 24 листопада, 2023 рік / ГО «Молодіжна наукова ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2023. – 298-300 сс.
97. Іваночко В. Р., Осолінський О.Р. «Алгоритм вимірювання часу обслуговування для SSD накопичувачів», Тренди та перспективи розвитку мультидисциплінарних досліджень: матеріали IV Міжнародної студентської наукової конференції, м. Луцьк, 1 грудня, 2023 рік / ГО «Молодіжна наукова ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2023. - 302-305 сс.

ДОДАТОК А

Блок-схема реалізації чорної скриньки



ДОДАТОК Б
КОД ПРОГРАМИ ТЕСТУВАННЯ ДЛЯ LINUX

```c
#include <stdio.h>
#include <stdlib.h>
// Константи для енергоспоживання (гіпотетичні значення)
#define IDLE_ENERGY 3.0 // Енергія, що споживається під час простою (у ват-годинах)
#define READ_ENERGY 5.0 // Енергія, що споживається за одну операцію читання (у ват-годинах)
#define WRITE_ENERGY 7.0 // Енергія, що споживається за операцію запису (у ват-годинах)

// Функція для імітації операції читання з різними розмірами блоків
void simulateReadOperations(int blockSizes[], int numBlocks) {
    for (int i = 0; i < numBlocks; ++i) {
        // Змоделювати операцію читання на основі розмірів блоків
        // (У реальному сценарії це буде доступ до блоків даних з жорсткого диску)
        int blockSize = blockSizes[i];
        printf("Читання блоку розміром %d байт\n", blockSize);
        // Змоделювати споживання енергії на операцію читання
        // (У реальному сценарії для цього потрібно виміряти енергоспоживання)
        printf("Спожито енергії: %.2f ват-годин\n", READ_ENERGY);
    }
}
// Функція для імітації операції запису з різними розмірами блоків
void simulateWriteOperations(int blockSizes[], int numBlocks) {
    for (int i = 0; i < numBlocks; ++i) {
        // Змоделювати операцію запису на основі розмірів блоків
        // (У реальному сценарії це буде запис блоків даних на жорсткий диск)
        int blockSize = blockSizes[i];
        printf("Запис блоку розміром %d байт\n", blockSize);
        // Змоделювати споживання енергії на операцію запису
        // (У реальному сценарії для цього потрібно виміряти енергоспоживання)
        printf("Спожито енергії: %.2f ват-годин\n", WRITE_ENERGY);
    }
}
int main() {
    int blockSizes[] = {512, 1024, 2048, 4096}; // Різні розміри блоків для операцій
    int numBlocks = sizeof(blockSizes) / sizeof(blockSizes[0]); // Кількість розмірів блоків

    // Імітація споживання енергії в режимі очікування
    printf("HDD у стані простою...\n");
    printf("Спожито енергії: %.2f ват-годин\n", IDLE_ENERGY);

    // Імітація операцій читання з різними розмірами блоків
    printf("\nІмітація операцій читання...\n");
    simulateReadOperations(blockSizes, numBlocks);

    // Імітація операцій запису з різними розмірами блоків
    printf("\nІмітація операцій запису...\n");
    simulateWriteOperations(blockSizes, numBlocks);

    return 0;
}
```

ДОДАТОК В
АПРОБАЦІЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Іваночко Вячеслав Романович, Магістр спеціальності 122 "Комп'ютерні науки"
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович, канд. техн. наук, доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

МЕТОД МОДЕЛЮВАННЯ "ЧОРНОЇ СКРИНЬКИ" ДЛЯ ПРИСТРОЇВ ЗБЕРІГАННЯ ДАНИХ

Вступ

Протягом багатьох років покращення продуктивності підсистеми вводу/виводу та інших підсистем комп'ютера відбувалося різними темпами, причому темпи покращення продуктивності підсистеми вводу/виводу були меншими за темпи покращення швидкодії підсистеми вводу/виводу, що ставило загальну продуктивність системи в залежність від швидкості підсистеми вводу/виводу.

Одним з найпоширеніших методів оцінки продуктивності підсистеми вводу/виводу комп'ютера є використання детальних імітаційних моделей, що включають специфічні особливості пристроїв зберігання даних, такі, як геометрія диска, розбиття на зони, кешування, буфери попереднього зчитування та переупорядкування запитів. Однак, як тільки з'являється нова технологічна інновація, ці моделі потрібно переробляти.

Альтернативним підходом є моделювання пристрою зберігання даних, як імовірнісної моделі "чорної скриньки".

Традиційно накопичувачі моделювали за допомогою детальних аналітичних моделей на основі геометрії пристроїв [1] або розбиття на зони [2, 3], які еволюціонували у багатьох випадках до рівня емуляторів.

Тим не менше, більшість цих моделей базуються на застарілих даних.

Якщо оцінювати масштабовані системи зберігання даних, то необхідна обчислювальна потужність зростає зі збільшенням кількості накопичувачів, які включені в модельовану систему.

Моделі "чорних скриньок"

Моделі "чорної скриньки" припускають, що про пристрій зберігання даних майже нічого не відомо. Пристрої та всі їхні характеристики розглядаються, як чорні скриньки, де внутрішні деталі та структура невідомі. Серед їхніх переваг можна назвати те, що вони, як правило, швидкі, оскільки не дуже деталізовані, тому їх легко конструювати і оновлювати.

Найпростішими моделями "чорних скриньок" є моделі на основі таблиць [4]. Вони просто пов'язують у таблицях пам'яті інформацію про вхідні дані від робочих навантажень з вихідними даними про реальний час обслуговування.

Чим більші таблиці, тим точніші моделі. Точність можна підвищити, додаючи нову інформацію до таблиць.

В роботі [5] пропонується вдосконалення роботи в [6]. Автори не використовують таблиці, а застосовують CART (Classification And Regression Tree) [7]. Автори роботи [8] оцінюють точність моделі чорного ящика для дисків на основі дерев регресії. Вони використовують алгоритм з відкритим вихідним кодом під назвою GUIDE [9]. Оцінки включають два типи середовищ:

– середовище з одним робочим навантаженням;
– середовище з декількома робочими навантаженнями.

У [10] продуктивність пристрою зберігання даних моделюється шляхом підстроювання характеристик продуктивності, використання ресурсів і робочого навантаження іншого пристрою, на якому базується перший пристрій.

Пропонований метод

Запропонований підхід реалізується наступним чином, який починається з послідовності запитів (від «синтетичного» робочого навантаження або реальних репозиторіїв трас) і закінчується генератором навантажень. Метод складається з декількох кроків, як показано на рисунку 1:

1.- Отримання послідовностей запитів на ввід/вивід
2 - Вимірювання часу обслуговування
3 - Побудова генератора навантажень
4 - Отримання результатів моделювання

Перший набір даних (крок 1) - це послідовність запитів вводу/виводу, яка використовується для вимірювання часу обслуговування диска:

- Реальні траси вводу/виводу з конкретної системи.
- Синтетичні робочі навантаження, що моделюють набір реалізацій трас вводу/виводу.

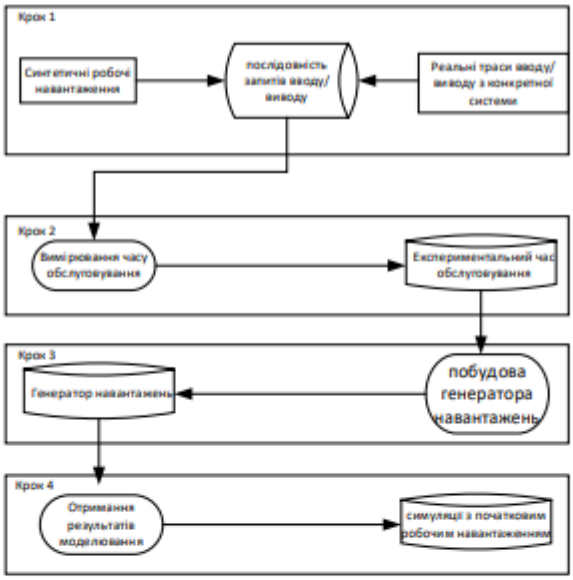


Рис. 1. **Етапи моделювання "чорної скриньки" пристроїв зберігання даних**

Вибір джерела для запитів вводу/виводу залежить від призначення симулятора, що створюється. Пропонований метод дозволяє працювати, як з реальними трасами вводу/виводу, так і з синтетичними робочими навантаженнями.

Модуль симуляції залежить від наборів даних запитів вводу/виводу. Якщо використовуються набори даних про запити вводу/виводу, що надходять з веб-сервера, модуль буде точно імітувати час обслуговування диска.

Наступним кроком (крок 2) є отримання наборів експериментальних трас з вимірами часу обслуговування диска.

Після вимірювання отримані дані аналізуються для побудови генератора навантажень (крок 3).

Генератор навантажень включається в модуль моделювання (крок 4) для отримання результатів моделювання.

Висновки

В даній роботі було описано метод побудови моделі чорної скриньки для накопичувачів і може бути використаний, як для синтетичних, так і для реальних дата-грам.

Метод включає в себе вимірювання часу обслуговування, яка може бути використана для накопичувача, з будь-яким типом інтерфейсу.

Для побудови моделей запропоновано два підходи. Перший підхід базується на декількох реальних трасах. Другий підхід базується на одній синтетичній трасі. Цей підхід обчислює час очікування в черзі «на льоту», на етапі прогнозування, що робить його більш універсальним.

Метод, описаний в цій роботі, забезпечує точний і більш загальний альтернативний підхід для моделювання часу обслуговування з будь-якого пристрою, з будь-яким типом інтерфейсу. Більша точність на відміну від інших підходів [11] забезпечується тим, що вимірювання часу обслуговування використовує стандартні виклики POSIX. Також запропонований метод може забезпечувати кращу продуктивність, що робить його доступним для моделювання в масштабах всієї системи.

Список використаних джерел:

1. AKGUN, Ibrahim Umit, et al. Improving Storage Systems Using Machine Learning. ACM Transactions on Storage, 2023, 19.1: 1-30.
2. LLOPIS, Pablo, et al. Model-based energy-aware data movement optimization in the storage I/O stack. The Journal of Supercomputing, 2017, 73: 5465-5495.
3. The DiskSim Simulation Environment (V4.0), 2008. URL: https://www.pdl.cmu.edu/DiskSim/index.shtml.
4. PAPON, Tarikul Islam; ATHANASSOULIS, Manos. A parametric I/O model for modern storage devices. In: Proceedings of the 17th International Workshop on Data Management on New Hardware (DaMoN 2021). 2021. p. 1-11.
5. XIE, Bing, et al. Predicting output performance of a petascale supercomputer. In: Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing. 2017. p. 181-192.
6. KARNIAVOURA, Flora; MAGOUTIS, Kostas. Decision-making approaches for performance QoS in distributed storage systems: A survey. IEEE Transactions on Parallel and Distributed Systems, 2019, 30.8: 1906-1919.
7. CARRIZOSA, Emilio; MOLERO-RÍO, Cristina; ROMERO MORALES, Dolores. Mathematical optimization in classification and regression trees. Top, 2021, 29.1: 5-33.
8. KUNJIR, Mayuresh; BABU, Shivnath. Black or white? how to develop an autotuner for memory-based analytics. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. 2020. p. 1667-1683.
9. YANG, Lingjian, et al. A regression tree approach using mathematical programming. Expert Systems with Applications, 2017, 78: 347-357.
10. CHEN, Feng; HOU, Binbing; LEE, Rubao. Internal parallelism of flash memory-based solid-state drives. ACM Transactions on Storage (TOS), 2016, 12.3: 1-39.
11. DIXTRAC: Automated Disk Drive Characterization, URL: http://www.pdl.cmu.edu/Dixtrac/index.shtml.

МАТЕРІАЛИ IV МІЖНАРОДНОЇ СТУДЕНТСЬКОЇ НАУКОВОЇ

КОНФЕРЕНЦІЇ

ТРЕНДИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ МУЛЬТИДИСЦИП-ЛІНАРНИХ ДОСЛІДЖЕНЬ

М. ЛУЦЬК, УКРАЇНА

1 ГРУДНЯ 2023 РІК

СЕКЦІЯ 17.
КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ

СЕКЦІЯ 18.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

СЕКЦІЯ 18.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

Іваночко Вячеслав Романович, магістр спеціальності 122 "Комп'ютерні науки"
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович, канд. техн. наук, доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

АЛГОРИТМ ВИМІРЮВАННЯ ЧАСУ ОБСЛУГОВУВАННЯ ДЛЯ SSD НАКОПИЧУВАЧІВ

Вступ

Надійність системи зберігання даних має важливе значення, оскільки збої в роботі компонентів системи зберігання можуть призвести до пошкодження або навіть повної втрати даних. Тому, окрім багаторівневого резервування, у дата-центрах поширеною практикою є своєчасна заміна пристроїв зберігання даних.

Прямим наслідком недотримання цього є витрати на заміну обладнання. Непрямим наслідком є час простою для усунення проблеми та/або заміни пристрою. Ремонт/заміна компонента системи зберігання даних після його виходу з ладу може зайняти навіть кілька днів, а пов'язаний з ним сервер протягом цього часу не працюватиме. Щоб компенсувати цей час простою, центри обробки даних вдаються до надмірного резервування (що може призвести до значних витрат).

У стеку систем зберігання даних твердотільні накопичувачі мають очевидну перевагу над жорсткими дисками з точки зору частоти збоїв. Однак (SSD коштують в 4-40 разів дорожче за ГБ, ніж жорсткі диски, залежно від класу (що нівелює і фактично переважує перевагу низької частоти відмов). Збій SSD в призводить до заміни в 79% випадків у порівнянні з 11% для збоїв пов'язаних з жорсткими дисками. Ці фактори, а також швидке поширення SSD[1, 2], відкривають поле для пошуку методів підвищення надійності SSD.

Поточні дані про частоту відмов твердотільних накопичувачів в основному надаються виробниками і базуються на прискорених лабораторних випробуваннях в контрольованих умовах.

На додаток до параметрів, на які вони тестуються, впливають інші фактори виробничого середовища (наприклад, різноманітні набори робочих навантажень, навколишнє середовище, політика управління тощо) які можуть бути не враховані. Крім того, простого розуміння частоти відмов, вказаної виробником, може бути недостатньо, навіть якщо вона відповідає дійсності.

Оператору центру обробки даних може знадобитися розуміння того, що, коли і чому відбувається, для прийняття відповідних рішень щодо резервування та

експлуатації, які нелегко відобразити за допомогою одного показника (частоти відмов).

Виходячи з даних аналізу моделювання робочих навантажень та вимірювання часу обслуговування є актуальною задачею. В даній роботі представлено алгоритм вимірювання часу обслуговування для SSD накопичувачів.

Вимірювання часу обслуговування

Після того, як робоче навантаження доступне, наступним кроком є вимірювання часу відповіді для кожного запиту. Важливо зазначити, що метою тут є не отримання загальної метрики (наприклад, середнього часу відгуку), а отримання вимірів для кожного запиту.

Робоче навантаження - це файл, який містить рядки запитів вводу/виводу. Кожен рядок містить інформацію про один запит, такий як адреса диска, на який він спрямований, його розмір, тип операції та часову мітку, коли запит повинен бути запущений і при оцінці продуктивності отримується робоче навантаження. На виході генерується файл з часом відгуку.

Для отримання робочого навантаження було використано програмний продукт DIXtrac, а саме утиліта даного пакету dxreplay [3, 4, 5]. Маючи певне робоче навантаження, dxreplay виконує вимірювання часу відгуку на накопичувачах і порівнює їх з вимірами, зробленими за допомогою Disksim [3]. Це робиться для перевірки правильності вилучення параметрів, які відповідають детальній моделі реального накопичувача у DiskSim. Він складається з головного потоку, який створює три екземпляри трьох потоків: lbnreader, issuer та collector. lbnreader читає запити з вхідного файлу робочого навантаження, issuer запускає раніше прочитані запити на диск у свою часову мітку, а collector чекає на завершення запущених запитів. Всі три потоки виконуються по черзі і синхронізуються за допомогою м'ютексів.

Програма проводить вимірювання за допомогою стандартних викликів POSIX, що складається з головного потоку, який зчитує кожен запит від робочого навантаження і запускає його у вказаний момент часу (алгоритм, рисунок 1). Він також містить обробник сигналів, який вмикається, коли певний запит завершився, і час його відповіді може бути переданий безпосередньо у вихідний файл, при виконанні налагоджувальних завдань або записаний у структуру даних в пам'яті (алгоритм, рисунок 2).

Кожна операція з робочого навантаження видається у вказаний час початку і для вимірювання часу відгуку. Коли запит завершено, час обслуговування записується у структуру даних у пам'яті, яка скидається при завершенні програми.

```
1. активація обробника сигналу
2. завантаження запиту в пам'ять з файлу робочого
навантаження
3. цикл (і <= до «кінця файлу»)
4.    {
5.     записати час запуску запитів для і-го кроку
6.     завантажити запит на диск за його міткою часу для
       і-го кроку
7.     чекати на наступний запит
8.    }
9. записати час відповіді у вихідний файл
вихід
```

Рис. 1. **Алгоритм вимірювання за допомогою стандартних викликів POSIX**

```
1. фіксувати час завершення запиту (завершений запит)
2. обчислити час відповіді (завершений запит)
3. вивести час відповіді за структурою даних  (завершений запит)
```

Рис. 2. **Алгоритм обробника сигналів**

Для проведення вимірювань на накопичувачі з метою отримання часу обслуговування, було змонтовано диск, як додатковий диск у системі, що має основний диск, на якому розміщується операційна система, інструменти оцінювання та пов'язані з ними файли з трасами оцінювання.

Щоб уникнути будь-якого впливу на процедуру оцінювання з боку файлової системи та драйвера пристрою, а також для того, щоб гарантувати, що кожен запит на введення/виведення у робочому навантаженні фізично надсилається до оцінюваного диска, використано ОС GNU/LINUX і диск визначається, як символьний пристрій, визначивши символьний пристрій (за допомогою mknod) і зв'язавши його з диском, який буде оцінюватися (за допомогою raw) можна отримати прямий доступ за допомогою викликів POSIX для читання і запису.

Висновки

В даній роботі було досліджено важливість надійності систем зберігання даних та їх впливу на безперервну роботу. Було описано проблеми, що виникають при відмові компонентів, які можуть призвести до втрати даних і значного зниження продуктивності. Також в даній роботі приведено наслідки збоїв накопичувачів пов'язані з ремонтом та заміною обладнання, а також важливість часу обслуговування для підвищення надійності накопичувачів.

Як виявилось, недостатня інформація від виробників та не врахування різних факторів може ускладнити розуміння реальної частоти відмов.

Аналізуючи методи вимірювання часу обслуговування для SSD накопичувачів, важливим моментом є те, що точність вимірювання для кожного запиту має велике значення. В роботі також наведено методи оцінки робочих навантажень та вимірювання часу обслуговування, як способи підвищення надійності систем зберігання даних.

Список використаних джерел:

1. WILKENING, Mark, et al. RecSSD: near data processing for solid state drive based recommendation inference. In: Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2021. p. 717-729.
2. MEI, Fei, et al. SifrDB: A unified solution for write-optimized key-value stores in large datacenter. In: Proceedings of the ACM Symposium on Cloud Computing. 2018. p. 477-489.
3. Steps to get disksim with dixtrac working for a 64bit machine, URL: https://github.com/dipietro-salvatore/disksim-64bit.
4. LI, Nanqinqin, et al. Fantastic SSD internals and how to learn and use them. In: Proceedings of the 15th ACM International Conference on Systems and Storage. 2022. p. 72-84.
5. THOMASIAN, Alexander. Mirrored and hybrid disk arrays: Organization, scheduling, reliability, and performance. arXiv preprint arXiv:1801.08873, 2018.