Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Дяченко М.М.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Дяченку Максиму Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 07.08.2024 року

3. Строк подання студентом роботи до захисту *2.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Маркетингове та економічне обґрунтування ІТ-проєкту.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська А.О.	05.10.2024	14.11.2024
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2024	16.11.2024

7. Дата видачі завдання « 6 » вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	2.12.2024 р.	

Дата видачі завдання
« 6 » вересня 2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Дяченко М.М. Дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи оцінки стану гібридних жорстких дисків SSD/HDD.

Метою розробки є дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD.

Об'єктом дослідження є процес оцінки стану гібридних жорстких дисків SSD/HDD.

Предметом дослідження є методи оцінки стану гібридних жорстких дисків SSD/HDD.

Методи дослідження базуються на методах комп'ютерної інженерії, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, гібридні жорсткі диски SSD/HDD

ABSTRACT

Dyachenko M.M. Research and software implementation of the SSD/HDD hybrid hard disk condition assessment system. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for the SSD/HDD hybrid hard disk condition assessment system.

The purpose of the development is the research and software implementation of the SSD/HDD hybrid hard disk condition assessment system.

The object of the research is the process of assessing the condition of SSD/HDD hybrid hard disks.

The subject of the research is methods for assessing the condition of SSD/HDD hybrid hard disks.

The research methods are based on computer engineering methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of the SSD/HDD hybrid hard disk condition assessment system.

In the process of working on the software model, an analysis of existing hardware and software tools was performed. All components of the developed software are fully described.

A user-friendly user interface has been developed. Instructions for working with the software are provided.

The program can be used on a PC with Windows 10/11.

The program was developed in the Python environment.

Keywords: computer engineering, hybrid hard drives SSD/HDD

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

API	–	прикладний програмний інтерфейс
HDD	–	жорсткий диск
LBA	–	адресація секторів
MFC	–	Microsoft Foundation Class library
RO	–	тільки читання
S.M.A.R.T.	–	Self-Monitoring, Analysis and Reporting Technology
БМГ	–	блок магнітних головок
ПЗ	–	програмне забезпечення

ВСТУП

Актуальність теми. У нашій роботі доводиться мати справу з різноманітними накопичувачами інформації, включаючи вінчестери й твердотільні накопичувачі інформації. При цьому іноді попадаються й досить незвичайні пристрої, які не поширені повсюдно. Наприклад, SSD/HDD – гібридні вінчестери.

Гібридні вінчестери, у першу чергу – компромісне рішення, що дозволяє як збільшити загальну продуктивність системи, у якій вони встановлені, так і знизити ціну такої системи.

Адже, незважаючи на широку поширеність, твердотільні накопичувачі ще досить дорогі, і навряд чи незабаром наступить момент, коли ціна таких накопичувачів значно впаде.

Звичайні жорсткі диски – недорогі, здебільшого, але їхня продуктивність обмежена, вище певного межі «стрибнути» не можна. Тому й з'явилися гібридні жорсткі диски. SSD/HDD з'явилися ще кілька років назад, і спочатку були чистої води екзотикою, що мало хто сприймав всерйоз (і мало хто знав про їх). Головним достоїнством гібридного жорсткого диска є збільшення загальної продуктивності системи, у якій вони встановлені, з використанням усього одного дискового відсіку (а не двох, якщо використовувати й SSD, і звичайного жорсткого диска). Зараз з'явилися моделі «гібридів» невеликих розмірів, наприклад, з товщиною всього в 7 мм (саме така модель ST500LM000 від Seagate), що дозволяє встановлювати такі диски в нетбуки/ультрабуки.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем оцінки стану гібридних жорстких дисків SSD/HDD.

– Дослідження системи оцінки стану гібридних жорстких дисків SSD/HDD.

– Програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD.

Об'єктом дослідження є процес оцінки стану гібридних жорстких дисків SSD/HDD.

Предметом дослідження є методи оцінки стану гібридних жорстких дисків SSD/HDD.

Методи дослідження базуються на методах комп'ютерної інженерії, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод оцінки стану гібридних жорстких дисків SSD/HDD.

– Розроблено вітчизняний продукт оцінки стану гібридних жорстких дисків SSD/HDD, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі оцінки стану гібридних жорстких дисків SSD/HDD.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2024 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Принцип роботи SSD/HDD ґрунтується на кешуванні найбільше часто використовуваних даних з використанням флеш-пам'яті, тобто SSD-частини «гібрида». Уже при першому запуску операційної системи на ноутбуці/ПК із «гібридом» в енергонезалежну частину пам'яті SSD/HDD містяться файли, які потрібні операційній системі для завантаження. У результаті швидкість запуску ОС збільшується, і досить значно. Гібридний диск, до речі, показує практично аналогічні результати швидкості передачі файлів у порівнянні зі звичайними жорсткими дисками. Але різниця в роботі різних типів пристроїв стає дуже помітною, якщо порівнювати час доступу до файлів (Access Time). Наприклад, якщо взяти диск Seagate ST500LT032 ємністю 500 ГБ і зрівняти з «гібридом» ST500LM000 аналогічної ємності, то швидкість доступу до файлів буде 24,2 і 0,3 мс. Що стосується граничність швидкості інтерфейсу, то різниця вже не в рази, а в 15%. У першому випадку 101 МБ/с, у другому – 115 МБ/с. Недоліки теж є, і в першу чергу, це – неможливість умістити всі критичні дані на SSD-частині SSD/HDD диска. Звичайно SSD в «гібриді» установлюється обсягом в 8 ГБ, іноді – більше (наприклад, нерідкі моделі з 32 ГБ флеш-пам'яті), але тоді такий диск буде вже дорожче. За ціною «гібриди» лише небагато перевищують ціну звичайних жорстких дисків. Якщо взяти вже згадувані вище моделі, то ціна Seagate Laptop Thin SSD/HDD ST500LM000 становить 73-75 доларів, а Seagate ST500LT032 – близько 50 доларів США. Так що, якщо ви хочете збільшити швидкість завантаження ОС, а також загальну продуктивність свого ноутбука/десктопного ПК, рекомендуємо використовувати «гібриди». Ну, це у випадку, якщо для вас важлива економія. Якщо немає – тоді варто використовувати SSD і звичайні вінчестери окремо.

1.2 Область застосування

Областю застосування є система перевірки на помилки, надійності та відказостійкості жорсткого диску SSD/HDD. Стан працездатності оцінюється по декількох параметрах роботи накопичувача, які називаються атрибутами надійності – attributes. Кожний атрибут має свій номер – ID (ідентифікатор). Атрибутам надійності відповідають параметри роботи накопичувача, які можуть характеризувати його природне зношування й передаварійний стан.

Більшість S.M.A.R.T. HDD мають від 3 до 15 атрибутів надійності. Максимально можлива їхня кількість 30. Состав і кількість атрибутів надійності визначаються самими виробниками індивідуально для кожного типу HDD.

Значення атрибутів надійності можуть лежати в діапазоні від 1 до 253. Спочатку атрибути мають максимальні значення. У міру виробітку ресурсу вінчестера або у випадку виникнення передаварійного стану значення атрибутів надійності зменшуються. Отже, високе значення атрибутів говорить про низьку ймовірність виходу накопичувача з ладу й, відповідно, низьке значення атрибутів – про низьку надійність накопичувача й високої ймовірності виходу його з ладу. Як правило, верхні границі атрибутів надійності мають значення 100 (IBM, Quantum, Fujitsu) або 253 (Samsung). Але є й виключення, так в HDD Western Digital моделей WDAC34000, WDAC33100, WDAC31600 перший атрибут надійності має максимальне значення 200, а інші 100.

Для кожного атрибута надійності розроблювачами HDD визначається граничне значення – threshold. Якщо хоча б одне зі значень атрибутів менше, ніж відповідне граничне значення, то зберігати дані на такому вінчестері стає небезпечно.

Значення thresholds залишаються постійними протягом всього життя накопичувача, а значення attributes зменшуються, наближаючись до граничного (thresholds).

Крім граничного значення для кожного атрибута визначений додатковий параметр pre-failure/advizory, так-же характеризує передаварійний стан

накопичувача. Можливі три стани накопичувача, характеризуємі станом pre-failure/advizory і значенням атрибута надійності:

– pre-failure/advizory = 0, характеризує високий запас надійності накопичувача за умови, що значення атрибута надійності більше відповідного граничного значення;

– pre-failure/advizory = 1, характеризує низький запас надійності накопичувача за умови, що значення атрибута надійності менше відповідного граничного значення;

– pre-failure/advizory = 2, характеризує передаварійний стан накопичувача за умови, що значення атрибута надійності менше відповідного граничного значення.

Всі S.M.A.R.T. параметри – attributes, thresholds і pre-failure/advizory зберігаються в енергонезалежній пам'яті HDD.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Гібридні жорсткі диски є проміжним рішенням між твердотільними накопичувачами й класичними HDD. За рахунок наявності додаткової Flash-пам'яті вони здатні буферізувати найбільше часто використовувані файли й згодом звертатися до них з набагато більше високою швидкістю. У цьому огляді ми познайомимо вас відразу із шістьома подібними пристроями.

Гібридні жорсткі диски (SSD/HDD – Solid State Hybrid Drive, твердотільний гібридний диск) важко назвати якоюсь інновацією й новинкою. По-перше, даний тип пристроїв уже досить давно випускається. Наприклад, розглянуті сьогодні моделі Laptop Thin SSD/HDD і Desktop SSD/HDD можна віднести до третього покоління SSD/HDD від Seagate. По-друге, принцип їхньої роботи досить простий. Це алгоритм Adaptive Memory, що записує найбільше часто використовувані дані у твердотільний кеш обсягом 8 Гбайт. Подібний принцип дозволяє помітно прискорити роботу операційної системи й часто використовуваних додатків. Це можуть бути ігри, офісні програми або ж графічні редактори. З обліком того, що ціна SSD/HDD не сильно вище, ніж ціна «звичайних» HDD, є зміст придивитися до цих рішень більш уважно – так ми й зробимо.

Гібридний жорсткий диск TOSHIBA

Із шести присланих до нас гібридних жорстких дисків тільки два накопичувачі виконані у форм-факторі 3,5 дюйми. Воно й зрозуміло: основну сферу використання подібного типу пристроїв – лептопи, тобто пристрою, у яких SATA-слот, як правило, один. Звичайно, усі бажають мати на борті ноутбука

SSD, але іноді дійсність виявляється куди-більше прозаїчніше: у першу чергу потрібен обсяг, а вже потім – висока швидкість. Десктопні 3,5-дюймові SSD/HDD користуються меншою популярністю, тому що персональний комп'ютер дуже часто дозволяє встановлювати відразу кілька накопичувачів різних форм-факторів.

«Гібриди» лінійки Laptop Thin SSD/HDD з'явилися ще торік. Це третьої покоління SSD/HDD Seagate, що замінило серію Momentus. Інші накопичувачі – абсолютні новинки цього року. До речі, поняття SSD/HDD першої в оберт увела саме компанія Seagate. Але тому що дана абревіатура не є товарним знаком, те подібне звертання до гібридних твердотільним вінчестерам почали використовувати й інші виробники.

Seagate Laptop Thin SSD/HDD

Лінійка Laptop Thin SSD/HDD нараховує відразу дві моделі обсягом 500 Гбайт і 1000 Гбайт відповідно. Від минулого покоління – Momentus – диски відрізняються зменшеною швидкістю обертання шпинделя (5400 про/хв проти 7200 про/хв), але збільшеним до 64 Мбайт кешем. Кількість Flash-пам'яті залишилося колишнім – 8 Гбайт.

Нарощування обсягу серед Laptop Thin SSD/HDD досягається шляхом збільшення числа млинців, тому накопичувачі мають різну товщину. Модель ST500LM000 обсягом 500 Гбайт може похвастати «стрункістю» – 7 мм. Отже, вона може бути інтегрована в абсолютну більшість лептопів. Більше ємне рішення за назвою ST1000LM014 має товщину 9,5 мм. Отут уже все залежить від конструкції ноутбука.

Seagate ST500LM000

За алгоритм Adaptive Memory відповідає спеціальна друкована плата. На ній розпаяно 8 Гбайт MLC-пам'яті Samsung K9LCGY8S1 B-HCK0. У ролі кеша виступає мікросхема Samsung K4T511630 J-BCE7. LSI 869002V0 – це контролер «гібрида».

Нижче наведені скріншоти з бенчмарка ATTO Disk Benchmark.

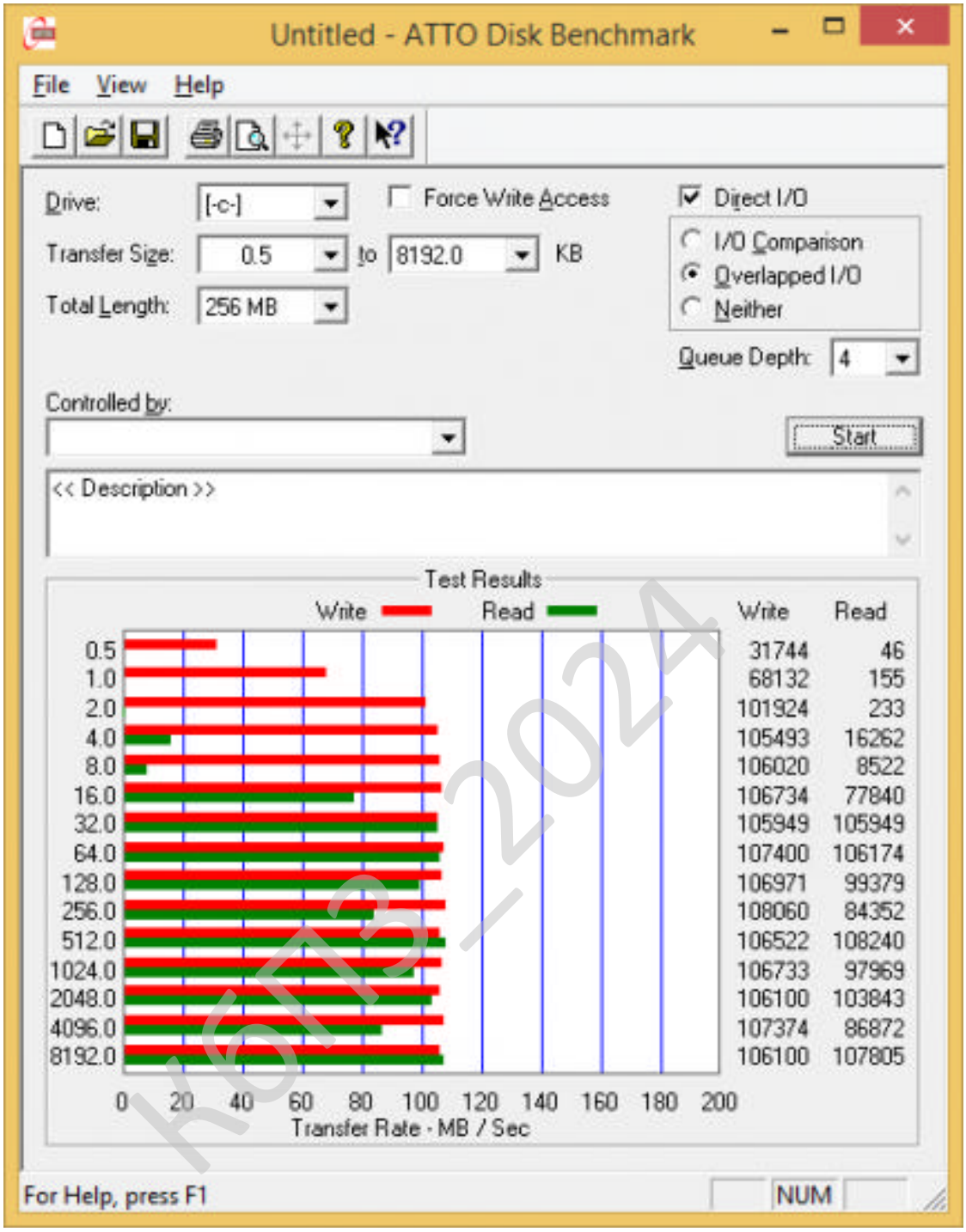


Рисунок 2.1 – Результати тестування Seagate ST500LM000, ATTO

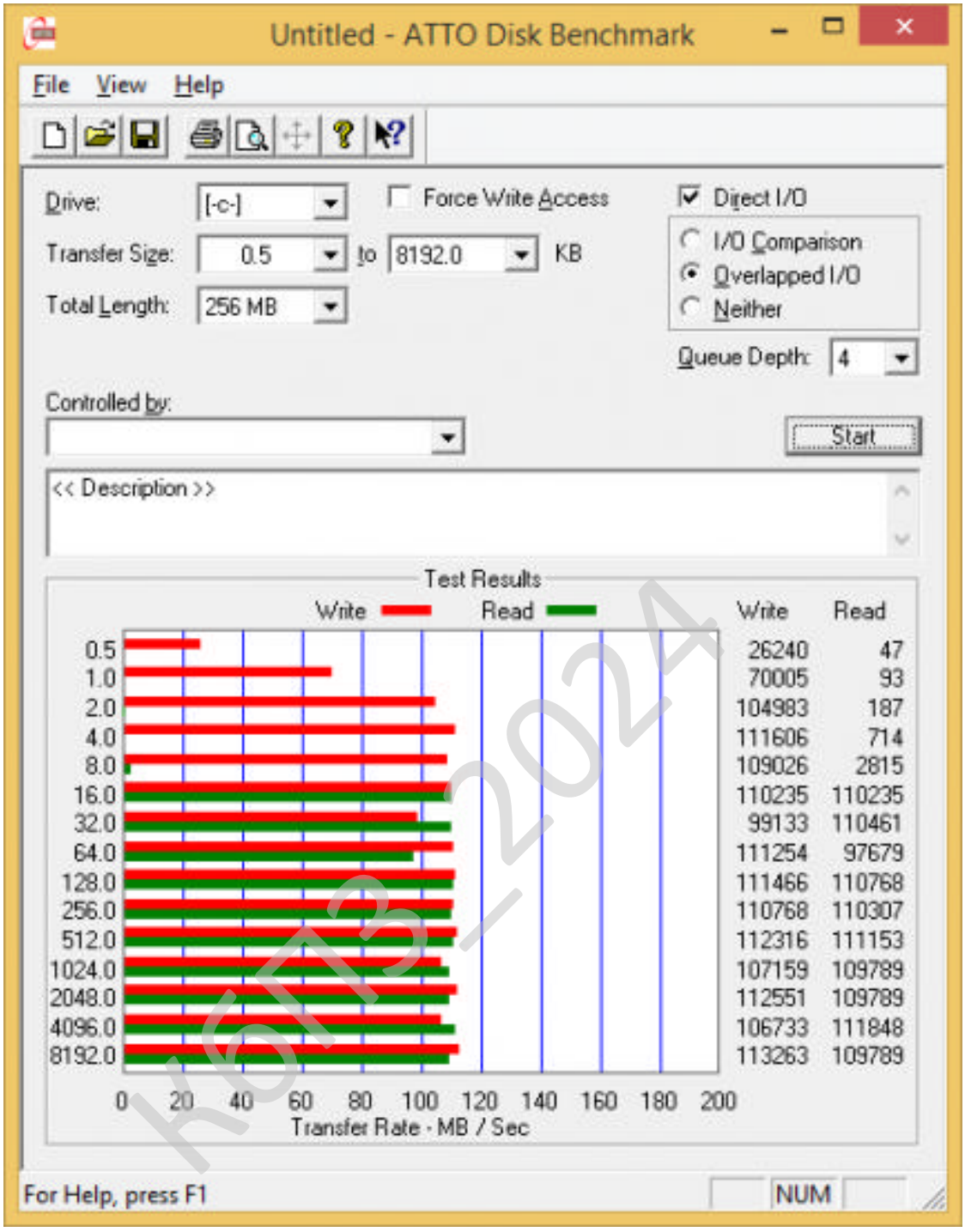


Рисунок 2.2 – Результати тестування Seagate ST1000LM014, ATTO

Seagate Desktop SSD/HDD

Лінійка пристроїв Desktop SSD/HDD складається із трьох моделей обсягом 1000 Гбайт, 2000 Гбайт і 4000 Гбайт відповідно. У перших двох дисків шпиндель здатний обертатися зі швидкістю 7200 про/хв. А от самий об'ємний гібридних

може похвастати показником частоти в розмірі 5900 про/хв. Позначається наявність великої кількості млинців. Відзначу, що SSD/HDD ємністю 1000 Гбайт має товщину корпуса 21 мм. Інші дві моделі – 26 мм.

Seagate ST2000DX001

У ролі Flash-пам'яті виступають мікросхеми компанії TOSHIBA, зроблені по 24-нм технологічних нормах. Використовуються два 64-гигабитних чипи, що з'єднуються через Toggle-Mode DDR 1.X. У свою чергу, це дає пропускну здатність на рівні 260 Мбайт/с. Однак виробник у технічних характеристиках заявляє піковий показник лінійного читання й запису в розмірі 190 Мбайт/с. Отже, 1000- і 2000-гігабайтні моделі впритул наближаються до жорстких дисків, шпиндель яких функціонує із частотою 10 000 про/хв. Буфер – мікросхема Samsung K4T511630 J-BCE7, тобто ті ж 64 Мбайт, що й в Laptop Thin SSD/HDD. Як контролер виступає LSI 869002V0.

Нижче наведені скріншоти ATTO Disk Benchmark. Як бачите, швидкість читання/запису моделі Seagate ST2000DX001 дійсно досягає заявлених 190 Мбайт/с.

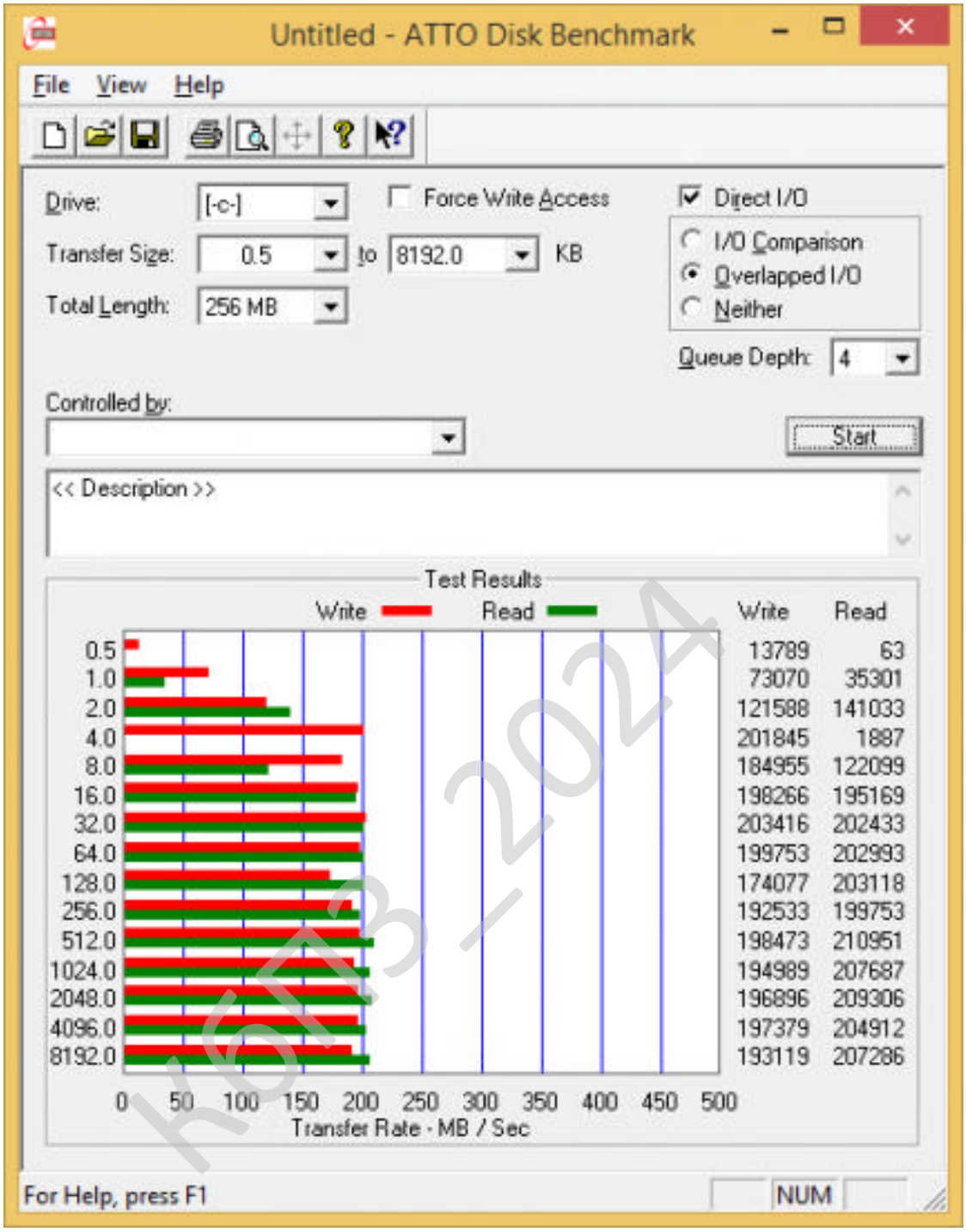


Рисунок 2.3 – Результати тестування Seagate ST2000DX001, ATTO

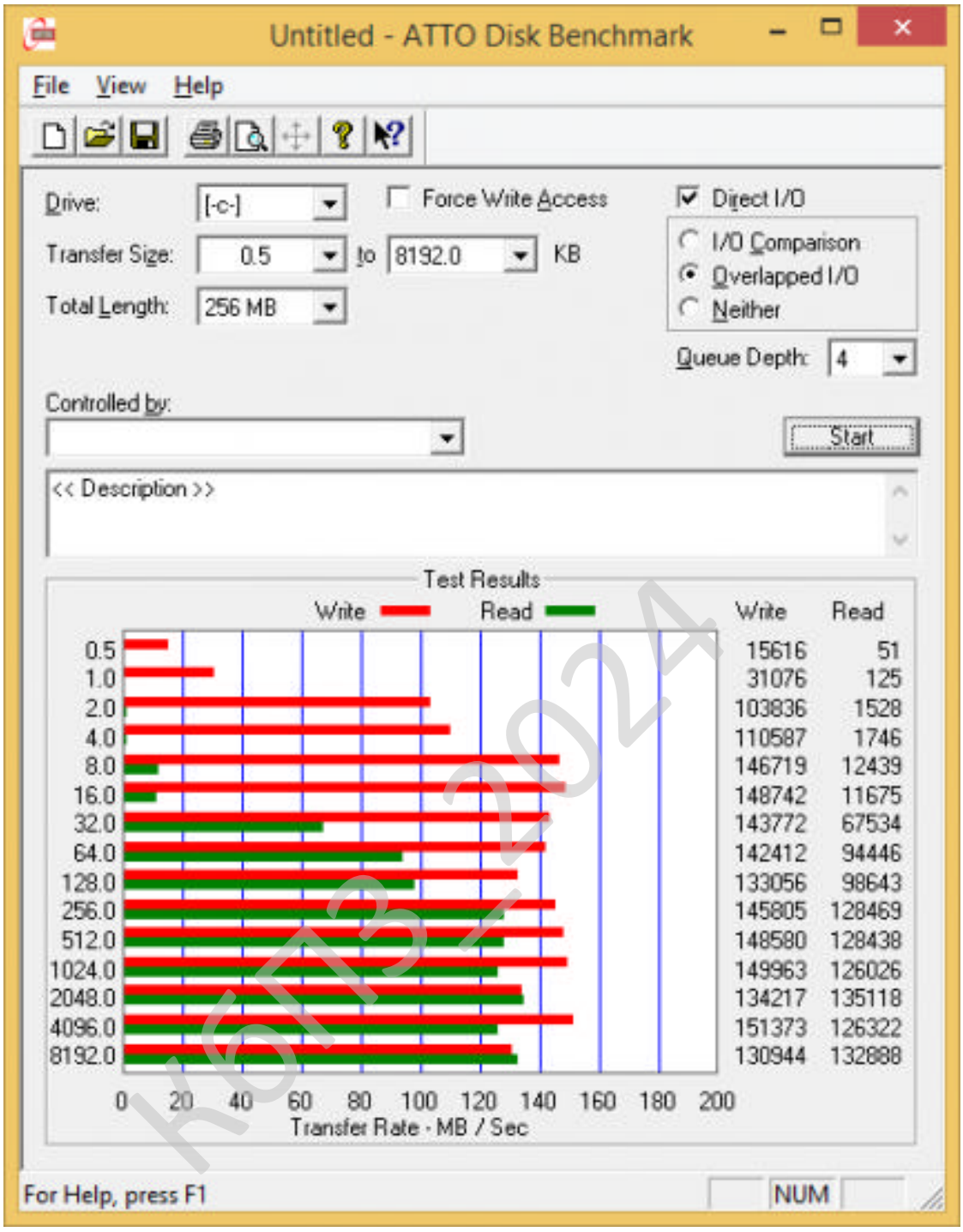


Рисунок 2.4 – Результати тестування Seagate ST4000DX001, ATTO

TOSHIBA MQ01ABFxxx/MQ01ABDxxx

Моделі MQ01AF050H і MQ01ABD100H представляють дві різні лінійки. Перша має 320-гігабайтну «напарницю», друга – 750-гігабайтну. Рішення MQ01AF050H може похвастати товщиною 7 мм. Друге – тільки 9,5 мм. Всі

SSD/HDD TOSHIBA оснащені 8 Гбайт Flash-пам'яті власного виробництва, а також 32 Мбайт кеша.

TOSHIBA MQ01AF050H

Marvell 88I 9317-RA12 – контролер безпосередньо HDD, а LINK-A-MEDIA 58NC7710GHB служить співпроцесором для гібридної частини. Також на платі розпаяна пам'ять TH58TEG6H2HBA4C.

Нижче наведені скріншоти ATTO Disk Benchmark. Як бачите, контролер Marvell відмінно працює із дрібними файлами, а от починаючи з 32 Кбайт швидкість читання й запису починає конкретно «гуляти». Однак у цілому видно, що послідовна обробка даних виробляється повільніше, ніж в SSD/HDD Seagate.

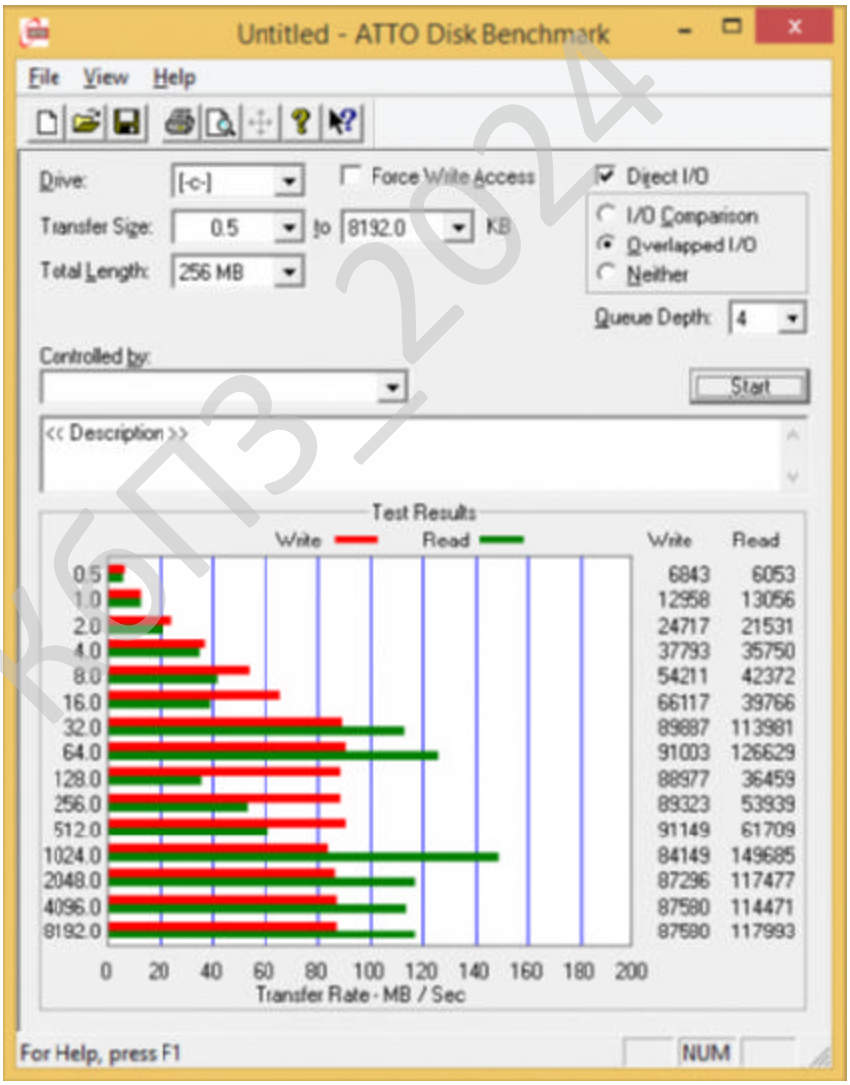


Рисунок 2.5 – Результати тестування TOSHIBA MQ01AF050H, ATTO

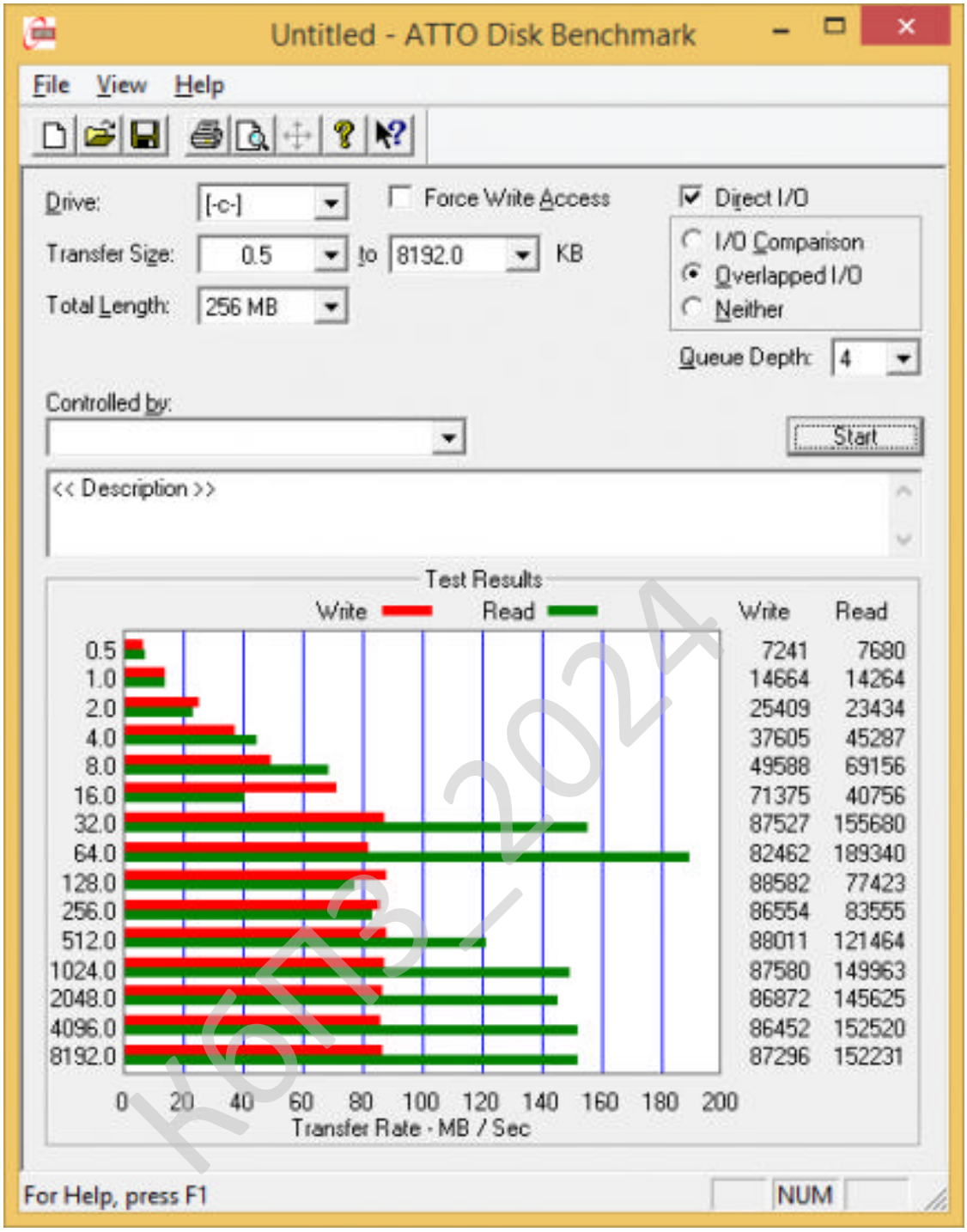


Рисунок 2.6 – Результати тестування TOSHIBA MQ01ABD100H, ATTO

Результати тестування

Тестовий стенд:

– Процесор: Intel Core i 5-4670K @4,5 ГГц.

– Процесорний кулер: Corsair H75.

– Материнська плата: ASUS Z 97-DELUXE.

– Блок живлення: LEPA G1600, 1600 Вт.

– Периферія: Samsung U28D590D, ROCCAT ARVO, ROCCAT SAVU.

– Операційна система: Windows 11 x64.

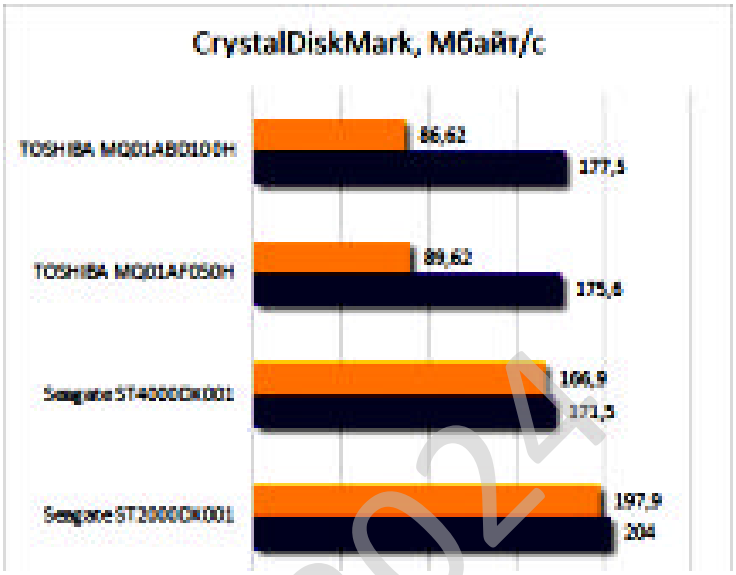


Рисунок 2.7 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

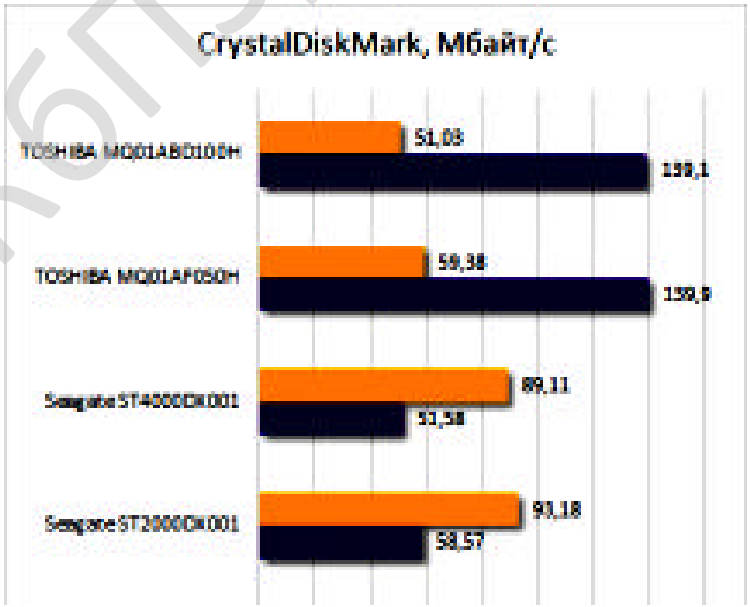


Рисунок 2.8 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

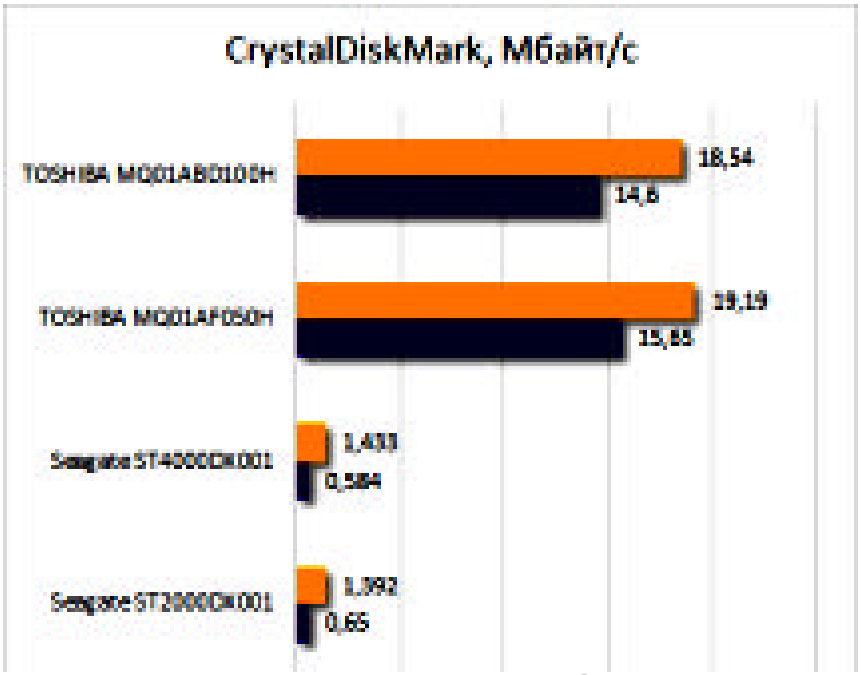


Рисунок 2.9 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

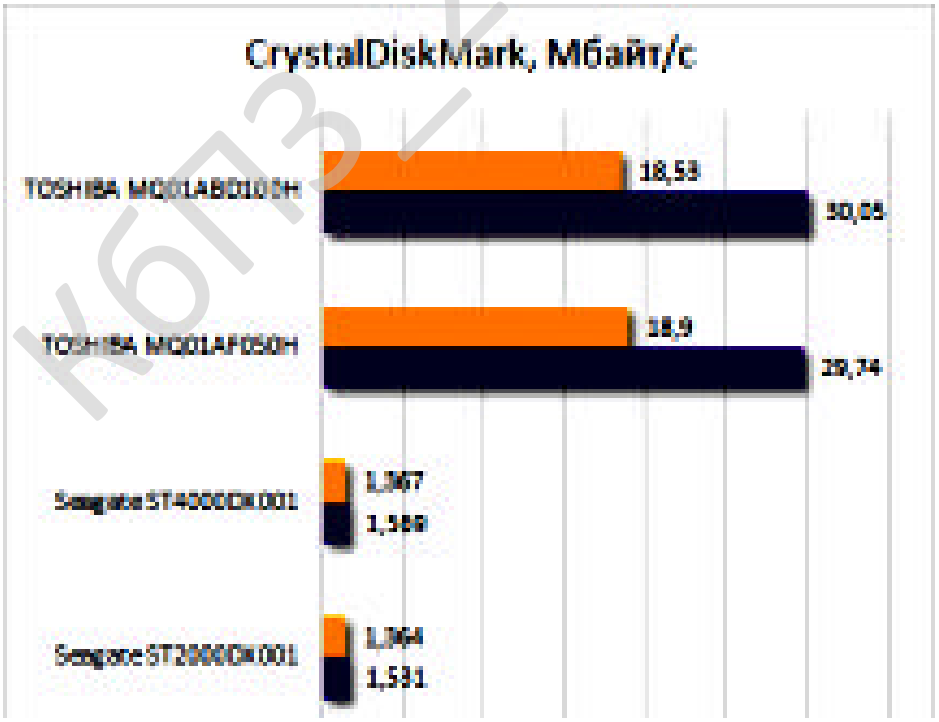


Рисунок 2.10 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

Почнемо з результатів CrystalDiskMark. Як я вже говорив, десктопна версія SSD/HDD обсягом 2000 Гбайт демонструє дуже високі швидкості. 204 Мбайт/с – це навіть вище, чим заявлено! При цьому й читання, і запис тримаються друг дружки, тобто ці операції виробляються з однаковою швидкістю. Також відзначу, що ST4000DX001 працює відчутно повільніше ST2000DX001, десь на 20-30 Мбайт/с.

Логічно, що 2,5-дюймові Laptop Thin SSD/HDD працюють помітно повільніше Desktop SSD/HDD Seagate. Також швидше працюють і «гвинти» від японського виробника.

У дисків TOSHIBA ситуація інша. Накопичувачі демонструють непогані швидкості читання – під 180 Мбайт/с. Але запис виробляється помітно повільніше й досягає максимальної оцінки 90 Мбайт/с.

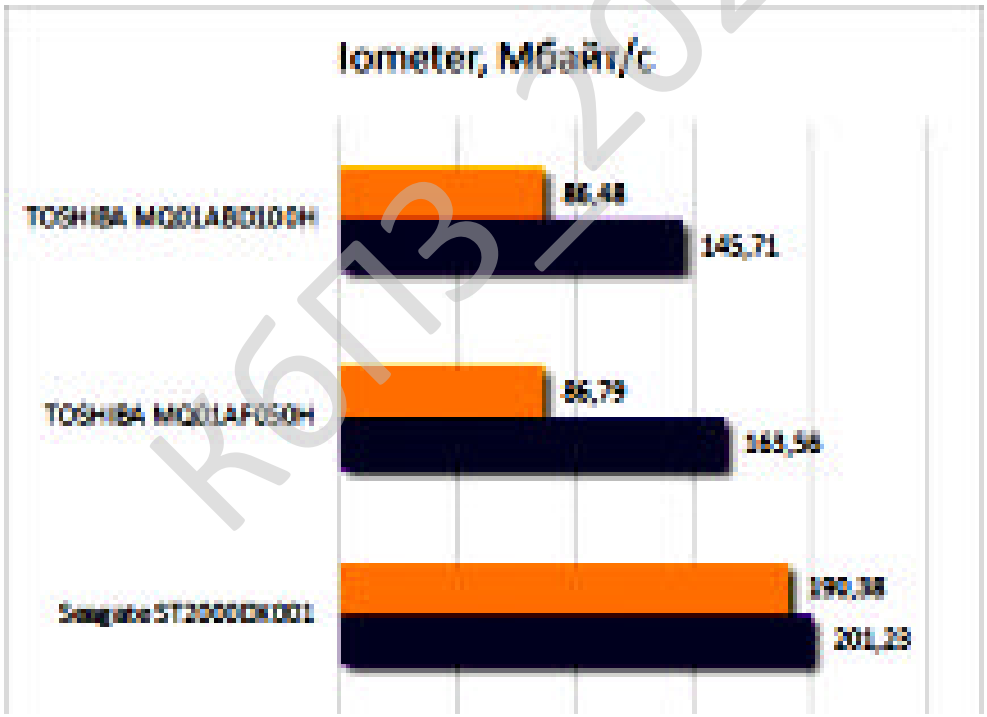


Рисунок 2.11 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

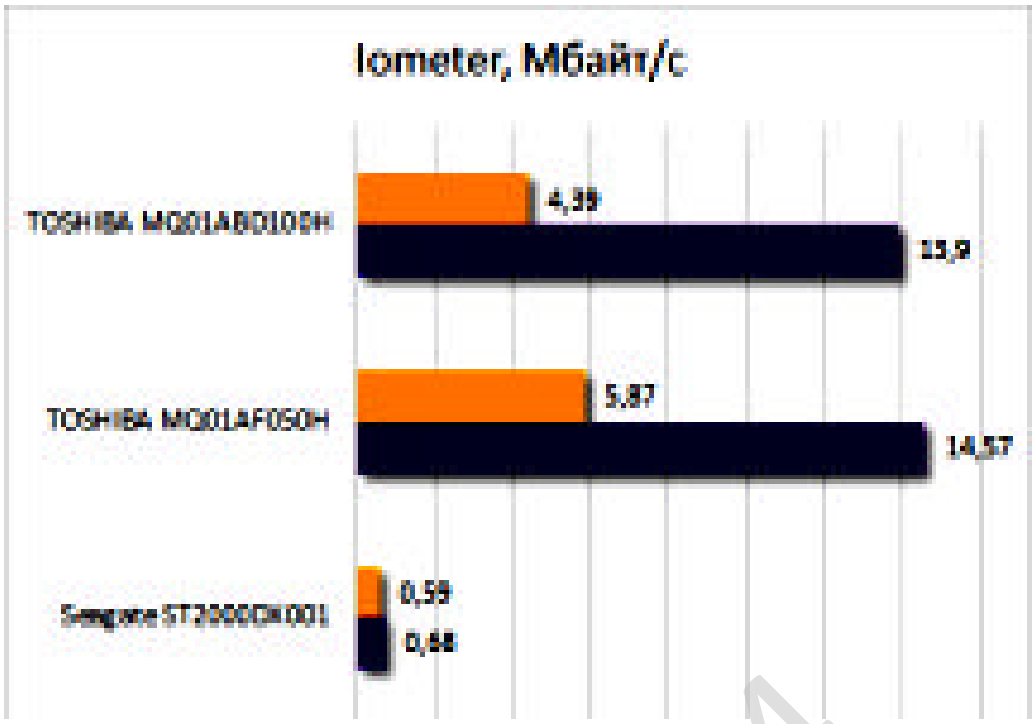


Рисунок 2.12 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

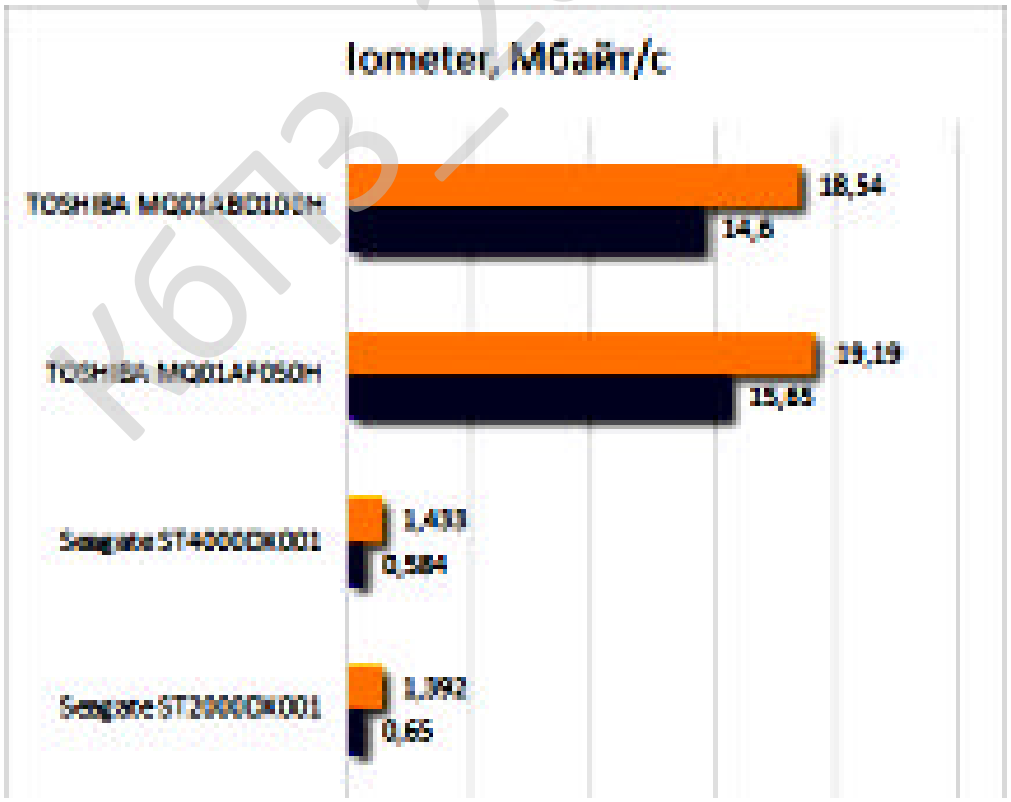


Рисунок 2.13 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

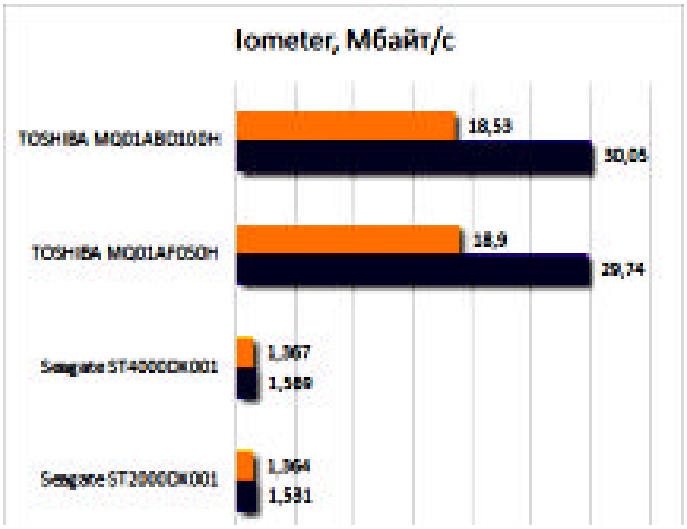


Рисунок 2.14 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

Класичні тести на послідовне/випадкове читання й запис в Iometer продемонстрували схожу тенденцію й аналогічний розклад сил. 2000-гігабайтний ST2000DX001 демонструє дуже гарна швидкодія. Але з файлами малого обсягу відверто пасує. Тут уже дуже пристойно виглядають SSD/HDD від TOSHIBA. Це відмінно показують паттерни Intel, що пристойно навантажують всю підсистему. Тут різниця у швидкості між «гібридниками» TOSHIBA і Seagate часом досягає 30-кратних величин.

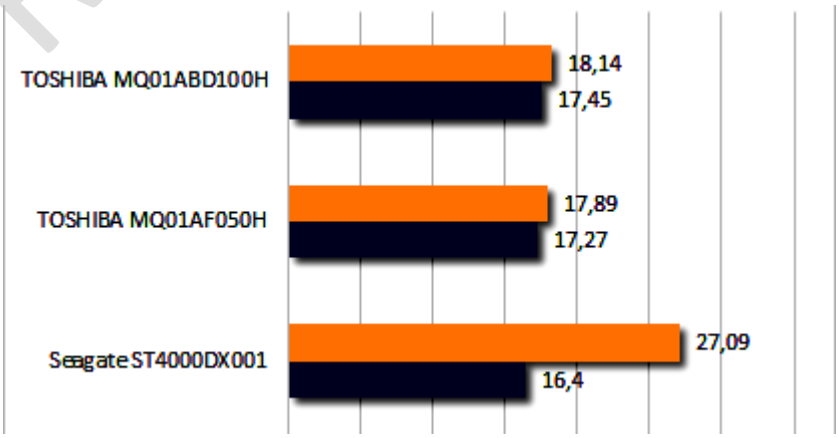


Рисунок 2.15 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

Устояний час відгуку у всіх дисків не представляє чого-небудь особливого й надприродного. Втім, цей час повинне компенсуватися наявністю Flash-пам'яті.

Для перевірки алгоритму Adaptive Memory ми використовували бенчмарк PCMark 7. У порівняльні таблиці занесені результати після одного й після п'яти прогонів підтеста System Storage.

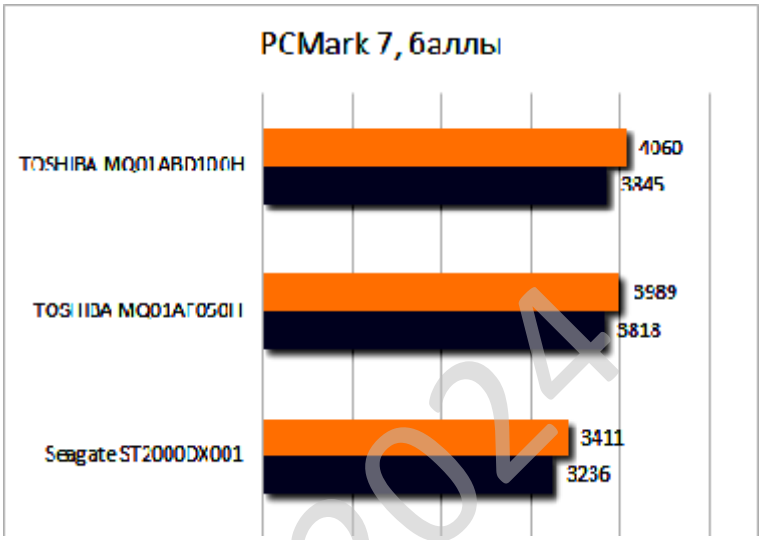


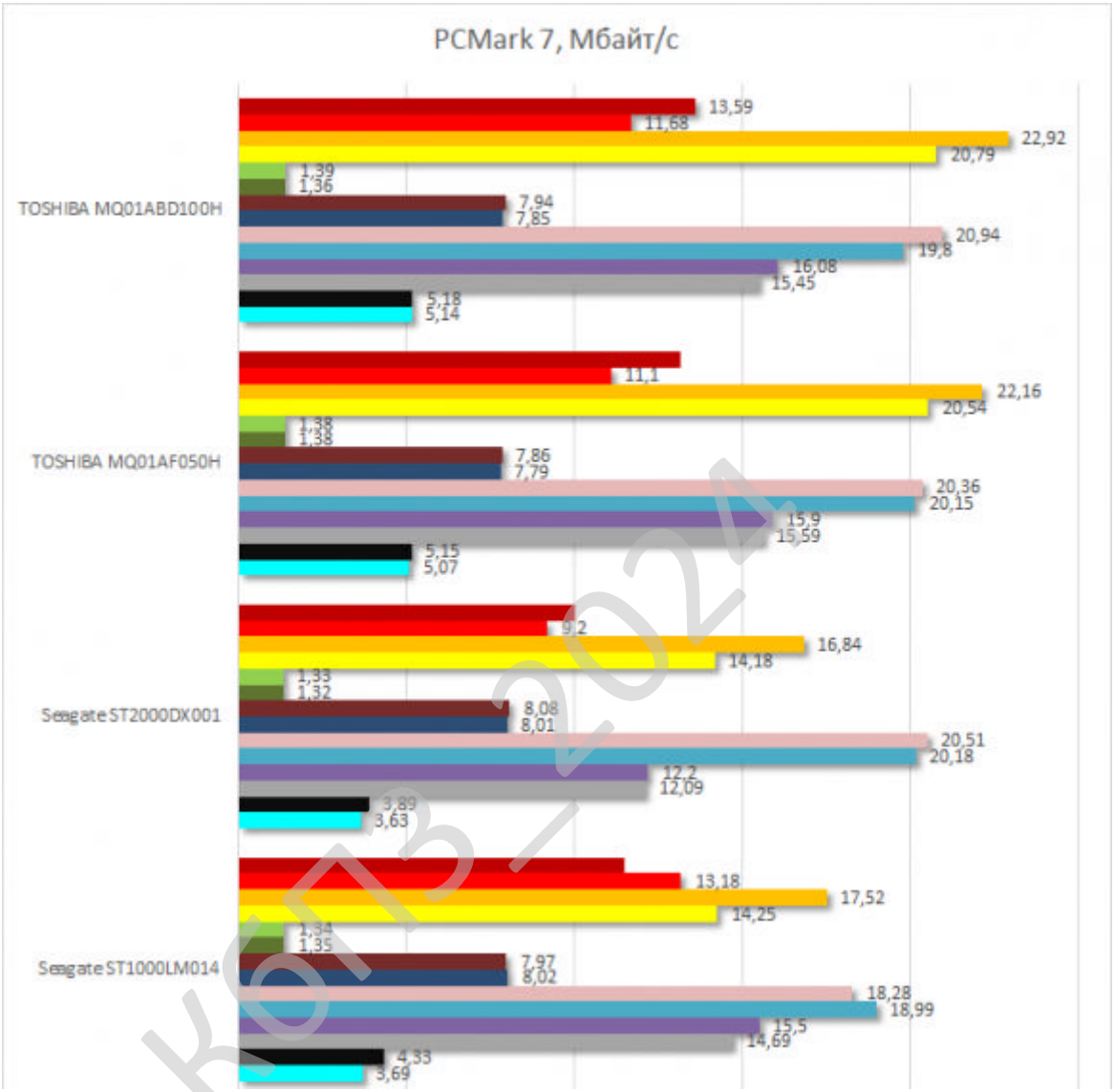
Рисунок 2.16 – Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

Результати виявилися очікуваними. Так, алгоритм Adaptive Memory працює. Однак конкретно в бенчмарке PCMark 7 приріст склав усього 5-7%. Отже, не варто чекати від SSD/HDD чуда. Часто використовувані додатки прискоряться, але все-таки зі швидкодією повноцінних SSD «гібридникам» ніколи не зрівнятися.

Конкуренти й аналоги

Як конкурент я вибрав дуже цікаву модель WD – Black2. Парадокс полягає в тім, що її можна назвати «гібридним твердотільним диском», але в той же час вона не є SSD/HDD. А всі тому, що WD Black2 – це пристрій два-в-одному. 9,5-міліметровий корпус форм-фактора 2,5-дюйми містить у собі повноцінний SSD

обсягом 120 Гбайт і повноцінний HDD обсягом 1000 Гбайт. При цьому пристрій оснащений усього одним рознімання SATA 3.0.



Результати тестування Seagate ST500LM000/ST1000LM014, Seagate ST2000DX001/ST4000DX001 і TOSHIBA MQ01AF050H/MQ01ABD100H

Висновки

Що ж, настав час підводити підсумки. Лінійка Laptop Thin SSD/HDD не показала нічого особливого. Використання шпинделя, що обертається зі швидкістю 5400 про/хв, помітно знизило швидкість роботи накопичувача. Диск

демонструє непогані результати при послідовному читанні й записі, хоча їм і далеко навіть до бюджетних моделей твердотільних накопичувачів. А от при випадковому читанні/запису, а також роботі із дрібними файлами SSD/HDD від Seagate буквально провалюється.

Набагато цікавіше, на мій погляд, виглядають SSD/HDD TOSHIBA. Наявність контролера Marvell дає про себе знати. Випадкове читання/запис, дані малого обсягу – у цих умовах і MQ01AF050H, і MQ01ABD100H демонструють відмінні показники швидкодії. Так, модель ємністю 1000 Гбайт (залежно від реселлера) на 500-1000 грн дорожче ST1000LM014, але воно того коштує.

Ще одна цікава модель – Seagate ST2000DX001. Накопичувач продемонстрував дужі швидкісні здатності при послідовному читанні й записі. Хоча робота з випадковими даними, а також файлами малих розмірів у порівнянні з тими ж SSD/HDD TOSHIBA залишає бажати кращого. Проте, якщо вам необхідний об'ємний накопичувач, але немає можливості встановити в десктоп SSD, те ST2000DX001 стане гарним вибором.

Модель Seagate ST4000DX001 по зрозумілих причинах уступає ST2000DX001, до того ж досить дорога.

Узагальнюючи, відзначу наступне:

– гібридні твердотільні жорсткі диски мають повні права на існування: при тім, що вони більш-менш порівнянні за ціною із класичними HDD, даний тип пристроїв дійсно прискорює роботу системи. Звичайно, не варто чекати від SSD/HDD спритності й ефективності SSD, однак найбільше часто використовувані програми будуть працювати швидше;

– все-таки, у наш час обсяг вирішує багато чого. Існує категорія користувачів (далеко не маленька, треба визнати), яких не влаштовує 120-256 Гбайт як дискова підсистема, реалізованої у вигляді SSD. Реалії ринку такі, що твердотільний накопичувач обсягом 1000 Гбайт обійдеться користувачеві в найкращому разі в 3-5 разів дорожче чим SSD/HDD. І це вагомий аргумент на користь гібридних вінчестерів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи оцінки стану гібридних жорстких дисків SSD/HDD.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи

технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Система призначена для реалізації моніторингу стану жорсткого диску SSD/HDD з використанням технології S.M.A.R.T.

S.M.A.R.T. (Self-Monitoring Analysis and Reporting Technology) – технологія самотестування, розроблена виробниками HDD для забезпечення більше високого ступеня надійності зберігання інформації. Суть S.M.A.R.T. технології полягає в тому, що вінчестер сам відслідковує стан своєї працездатності й здатний заздалегідь попередити користувача про свій передаварійний стан.

IT-інфраструктура пройшла довгий шлях. Галузь швидко розвивається, а безліч рішень стає дедалі складнішим. IT почали обслуговувати все більш різноманітні потреби бізнесу та технологій у всіх організаціях на планеті.

Як у гіперконвергентних, так і в традиційних рішеннях параметри зберігання даних зазнали багатьох цікавих змін. Від технології жорстких дисків ми все більше переходимо до набагато швидших твердотільних накопичувачів. Тепер покупці мають більше можливостей, ніж будь-коли раніше, коли вони обирають поєднання технологій для своєї IT-інфраструктури. Хоча жорсткі диски добре відомі та надійні, твердотільні накопичувачі стають все більш популярними та доступними. Існує також третій варіант, коли справа стосується зберігання даних IT-інфраструктури: гібридне сховище.

У цій статті описано, що передбачає концепція гібридного сховища, і чому ми вважаємо, що вона дуже добре підходить для підприємств будь-якого розміру та будь-якої галузі.

Що таке гібридне сховище?

Варіанти гібридного зберігання даних використовують комбінацію

жорстких дисків і твердотільних накопичувачів, що знижує загальну вартість рішення, а також забезпечує швидкість і ефективність флеш-пам'яті. Гібрид є дуже вдалим вибором, оскільки він пропонує «традиційну» надійність і низьку вартість жорстких дисків, а також підвищену продуктивність твердотільних накопичувачів в одному рішенні.

Звичайно, це поєднання низької вартості/надійності та підвищеної продуктивності працює лише в тому випадку, якщо мережа керування програмним забезпеченням достатньо розумна, щоб розумно розподіляти та розподіляти ресурси потужності; Скинути купу SSD і HDD в будь-який старий масив не вийде. Це причина, чому кілька постачальників флеш-пам'яті знайшли особливо сильний маркетинговий сигнал проти гібридних масивів. Насправді, якщо поєднання SSD/HDD правильно використано, будь-яка організація може використовувати переваги обох за значно менших витрат.

4 переваги гібридного сховища:

1. **Менші витрати**

Як згадувалося раніше, гібридне сховище значно доступніше, ніж флеш-флеш-альтернатива, водночас пропонуючи більшу продуктивність, ніж традиційні диски. Зазвичай це забезпечує дуже хороше співвідношення ціни та ефективності.

Перш ніж купувати повний спалах, ви повинні спочатку запитати себе, чи дійсно він вам потрібен. Інакше ви ризикуєте отримати надлишок ресурсів. Скільки IOPS вам насправді потрібно? Так, флеш-пам'ять може обробляти велику кількість IOPS, але чи це важливо для вашого бізнесу? Можливо, ви також обробляєте багато «холодних» даних, які можна комфортно зберігати на традиційних жорстких дисках? Якщо ви шукаєте комбінацію з кількох програм, які вимагають швидких твердотільних накопичувачів і великої кількості старих даних, які рідко використовуються, гібридне рішення є найкращим вибором.

2. **Підвищена продуктивність**

Незалежно від типу IT-інфраструктури, традиційної чи

гіперконвергентної, продуктивність ємності зберігання можна значно покращити, додавши трохи флеш-пам’яті до комбінації. У Syneto варіанти гібридної ємності в наших гіперконвергентних продуктах пропонують у 5-20 разів більшу продуктивність* порівняно з традиційним накопичувачем на основі жорсткого диска.

Програмно визначена інфраструктура є ключем до розумного використання кількох носіїв інформації. Це дозволяє ОС керувати набором пристроїв зберігання (DRAM, SSD, HDD тощо) в єдиному пулі даних.

3. **Підвищена ефективність**

Повільний доступ до даних для користувачів і програм може спричинити серйозні проблеми для бізнесу. Усунення несправностей і вирішення проблем продуктивності вимагає багато часу та зусиль.

Гібридні рішення для зберігання, керовані програмним рівнем, більш ефективні при управлінні «гарячими» даними. Програмне забезпечення оптимізує розміщення «гарячих» і «холодних» даних. Рідше та нещодавно використовувані дані зберігатимуться на жорстких дисках, тоді як часто та нещодавно використовувані дані оброблятимуться швидшим флеш-шаблоном.

4. **Спрощене управління**

Вибираючи програмно-визначену інфраструктуру, ви одразу отримуєте перевагу спрощеного керування. Значно скорочується час, витрачений на налаштування та керування ємністю зберігання вашого середовища. Інфраструктуру легко налаштувати та керувати нею, що дозволяє вашій IT-команді зосередитися на інших частинах бізнесу.

3.2 Розробка структурної схеми

На рисунку 3.1 зображена структурна схема розробленої систем оцінки стану гібридних жорстких дисків SSD/HDD.

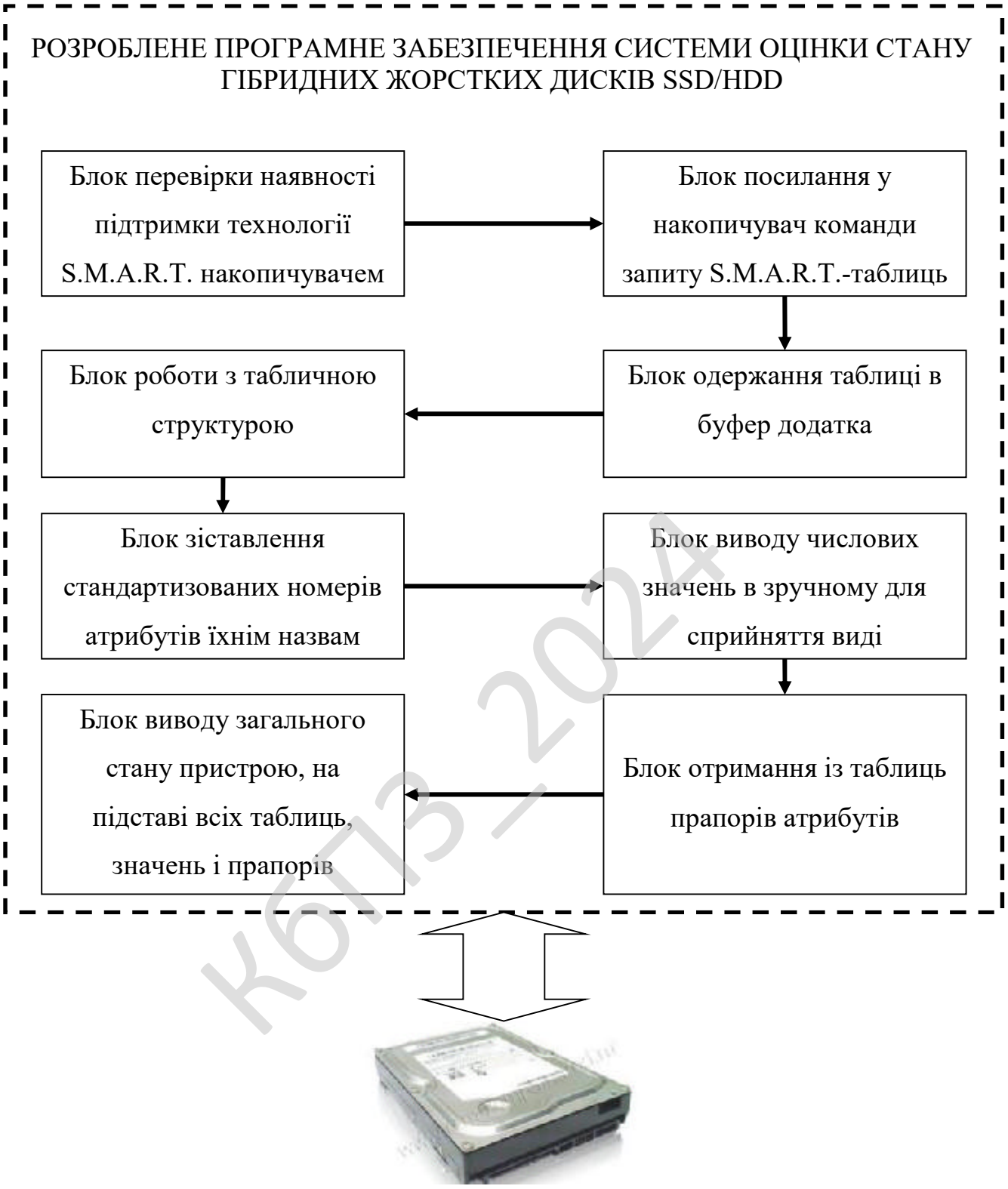


Рисунок 3.1 – Структурна схема системи

Структурна схема складається з наступних блоків:

– Блок перевірки наявності підтримки технології S.M.A.R.T. накопичувачем.

– Блок посилання у накопичувач команди запиту S.M.A.R.T.-таблиць.

– Блок одержання таблиці в буфер додатка.

– Блок роботи з табличною структурою, що витягає з них номера атрибутів і їхні числові значення.

– Блок зіставлення стандартизованих номерів атрибутів їхнім назвам (іноді – залежно від типу, моделі або фірми-виробника HDD).

– Блок виводу числових значень в зручному для сприйняття вигляді.

– Блок отримання із таблиць прапорів атрибутів (ознаки, що характеризують призначення атрибута в рамках конкретної firmware накопичувача, наприклад, «життєво важливий» або «лічильник»).

– Блок виводу загального стану пристрою, на підставі всіх таблиць, значень і прапорів.

У комп'ютерному сховищі в розпорядженні споживачів є різноманітні альтернативи, кожна з яких має свої переваги та недоліки. Двома з найбільш популярних варіантів є твердотільний накопичувач (SSD) і твердотільний гібридний накопичувач (SSHD). Цей дискурс заглиблюється в розбіжності між гібридним накопичувачем і SSD, щоб допомогти вам визначити оптимальний вибір для ваших потреб.

Що таке SSD?

Твердотільні накопичувачі створені з використанням передової флеш-пам'яті NAND і твердотільного накопичувача. Це надає їм можливість виконувати операції читання та запису даних зі значно прискореною швидкістю порівняно зі звичайними жорсткими дисками. Крім цієї підвищеної швидкості, твердотільні накопичувачі демонструють більшу енергоефективність, знижене тепловиділення та тихішу робочу акустику порівняно з традиційними жорсткими дисками. Зростання популярності твердотільних накопичувачів останнім часом

пояснюється їх високою продуктивністю, стійкістю та меншою чутливістю до втрати даних.

Твердотільна технологія додатково сприяє більш компактному фізичному відбитку, що робить їх придатними для інтеграції в ноутбуки. Тим не менш, важливо визнати, що твердотільні накопичувачі можуть мати вищу вартість, ніж їх звичайні аналоги жорстких дисків, і їх ємність зберігання може мати обмеження в порівнянні. Оцінюючи гібридний жорсткий диск проти SSD, слід ретельно зважити унікальні характеристики вашої операційної системи та моделі використання, як це прийнято для будь-якого технологічного рішення.

Як працює і як він використовується?

Функціонування SSD обертається навколо використання масиву взаємопов'язаних мікросхем пам'яті для розміщення даних із застосуванням процедури, яка називається флеш-пам'яттю. Ці мікросхеми пам'яті містять високопродуктивну флеш-пам'ять NAND, яка сприяє швидким і ефективним процедурам читання та запису.

Після збереження файлу на SSD дані розбираються на дрібні блоки та вписуються у вільні комірки пам'яті, розташовані всередині мікросхеми. Після запиту користувача на доступ до файлу SSD отримує відповідні дані з чіпів пам'яті та передає їх до центрального процесора комп'ютера для програми. Щоб скористатися SSD, необхідно встановити з'єднання між накопичувачем і комп'ютером через сумісний інтерфейс, наприклад SATA, NVMe або PCIe.

Після встановлення з'єднання SSD можна використовувати як будь-який звичайний накопичувач, полегшуючи збереження файлів і інсталяцію додатків звичайним способом. Проте варто підкреслити, що твердотільні накопичувачі мають обмежений робочий період і можуть пройти лише певну кількість циклів запису, перш ніж досягти кінця свого терміну служби.

Щоб оптимально подовжити термін служби вашого SSD, розважливий підхід вимагає розумного підходу до безперервного запису значних обсягів даних у поєднанні з активацією таких функцій, як TRIM. Ці заходи спільно

допомагають в управлінні зберіганням і підтримці рівня продуктивності протягом часу.

Що таке SSHD?

SSHD використовує функції, властиві високопродуктивній флеш-пам'яті NAND, щоб сприяти швидшому пошуку даних для часто використовуваних файлів. Ця технологічна структура дозволяє SSHD функціонувати подібно до стандартного жорсткого диска, водночас наділяючи його певними перевагами, що нагадують SSD.

SSHD вміло організовує переміщення файлів, до яких часто звертаються, у сегмент твердотільного накопичувача, забезпечуючи таким чином оперативний доступ, одночасно переміщуючи менш часто використовувані файли в звичайний сегмент жорсткого диска. Ця оркестровка регулюється мікропрограмою пристрою, яка точно визначає розміщення файлів у твердотільному компоненті. Інтеграція SSHD у ваш комп'ютер або пристрій так само проста, як встановлення традиційного жорсткого диска.

Операційна система та програми взаємодіють із SSHD так само, як із звичайним жорстким диском. Користувач звільнений від необхідності мікрокерування сховищем на SSHD, оскільки мікропрограмне забезпечення пристрою безперебійно керує міграцією даних між твердотільним і традиційним секторами зберігання. Єдина увага користувача залежить від сукупної ємності пристрою, що залежить від розмірів твердотільного та звичайного частин накопичувача.

Як функціонує SSHD і як його використовувати?

Прагнучи підвищити ефективність роботи SSHD, накопичувач використовує спеціальний алгоритм, розроблений для ретельного вивчення шаблонів файлів і програм, які часто використовуються. Згодом цей алгоритм автономно переносить ідентифіковані файли та додатки на твердотільний накопичувач, тим самим полегшуючи прискорений доступ.

З плином часу алгоритм отримує глибше розуміння даних, до яких

найчастіше звертаються, таким чином спонукаючи до динамічного повторного калібрування розміщення даних.

Відновлення даних з гібридних дисків

Використання SSHD точно віддзеркалює використання звичайного жорсткого диска. Акт збереження файлів на ньому, встановлення програм і доступ до збережених даних узгоджуються зі звичайними практиками. Примітно, що SSHD вирізняється автоматичним тонким налаштуванням загальної продуктивності за допомогою стратегічного зберігання даних, до яких часто звертаються, у твердотільному компоненті накопичувача.

Тим не менш, доцільно визнати, що ємність твердотільного накопичувача SSHD зазвичай не досягає ємності автономного SSD, що потенційно може спричинити різницю в рівнях продуктивності порівняно з повним рішенням SSD.

SSD проти SSHD – Основні відмінності між SSD та SSHD

Порівняння між гібридним SSHD і SSD висвітлює різні рішення для зберігання, кожне з яких надає різноманітні переваги залежно від вимог користувача. Накопичувачі SSD вирізняються швидкістю та надійністю в порівнянні з SSHD завдяки відсутності в них рухомих компонентів і використанню технології флеш-пам’яті. Така архітектурна композиція робить їх оптимальними для високопродуктивних видів діяльності, таких як ігри та редагування відео. Крім того, вони виявляють знижене енергоспоживання та виділення тепла, що є перевагою для ноутбуків та інших портативних пристроїв.

Навпаки, SSHD поєднують характеристику швидкодії SSD з розширеною ємністю зберігання, яка приписується традиційним жорстким дискам (HDD). Це злиття дає економічно ефективну альтернативу для користувачів, яким потрібна значна ємність без значних витрат на SSD. SSHD поєднують скромний розподіл флеш-пам’яті та більший компонент жорсткого диска для розміщення даних, до яких часто звертаються, що завершується швидшим завантаженням і швидшим завантаженням програм.

Під час обговорення між гібридними жорсткими дисками та SSD розумно

слід враховувати фінансові обмеження, вимоги до зберігання та спектр передбачуваних завдань. Твердотільні накопичувачі, хоча загалом і більш фінансово вимогливі, забезпечують чудову швидкість читання та запису, що є перевагою для ресурсомістких операцій. У той же час, SSHD, відзначені економічною доцільністю, пропонують розширену ємність зберігання, яка резонує з рутинними програмами.

Зрештою, перевага між SSD і гібридними жорсткими дисками залежить від ваших конкретних потреб і переваг. Обидва шляхи демонструють свої притаманні переваги та обмеження, таким чином покладаючи на вас обов’язок визначити оптимальну відповідність вашим обчислювальним вимогам. Завдяки інформованому оцінюванню відмінностей, що характеризують гібридний диск і SSD, ви готові зробити вибагливий вибір, таким чином прийнявши рішення для зберігання даних, яке відповідає вашим вимогам і бюджету.

3.3 Розробка функціональної схеми

Користувач комп'ютера, оснащеного S.M.A.R.T. і спеціальною програмою S.M.A.R.T. діагностики, буде заздалегідь знати про можливий передаварійний стан і, отже, зможе уникнути втрати даних які зберігаються на вінчестері. У цей час S.M.A.R.T. технологію підтримують всі виробники: Seagate, Western Digital, Quantum, Fujitsu, Maxtor, Samsung, Hitachi, IBM. При цьому потрібно також сказати, що технологія не в змозі пророчити абсолютно всі можливі проблеми й це логічно: вихід електроніки в результаті стрибка напруги, псування головок і поверхні в результаті удару й т.п. ніяка технологія пророчити не в силах. Передбачувані лише ті проблеми, які пов'язані з поступовим погіршенням яких-небудь характеристик, рівномірною деградацією яких або компонент.

S.M.A.R.T. являє собою набір міні-підпрограм, які є частиною мікрокоду накопичувача й визначають підтримувані діагностичні функції. Найпоширеніші серед них:

– набір атрибутів, що відбивають стан окремих параметрів накопичувача (до 30);

– внутрішні тести накопичувача (self– test);

– журнали S.M.A.R.T. (помилок, загального стану, дефектних секторів і т.п.).

У даний момент не існує офіційної документації або стандарту на технологію S.M.A.R.T. У зв'язку із цим, виробники не публікують повні характеристики й підтримувані функції S.M.A.R.T. у своїх накопичувачах. Обов'язковий мінімум описаний в останньому стандарті ATA/ATAPI-6.

Розвиток технології S.M.A.R.T.

Історія технології S.M.A.R.T. не так уже й багата подробицями:

– S.M.A.R.T. I передбачав моніторинг основних життєво важливих параметрів і запускався тільки після команди по інтерфейсу.

– в S.M.A.R.T. II з'явилася можливість фонової перевірки поверхні, що виконувалася накопичувачем автоматично під час "холостого ходу"; з'явилася функція журналювання помилок.

– в S.M.A.R.T. III уперше з'явилася не тільки функція виявлення дефектів поверхні, але й можливість їхнього відновлення "прозоро" для користувача й багато інших нововведень.

Відомо, що першими розробили основи й запропонували цю технологію спільно Western Digital, Seagate і Quantum. Після цього їх уже підтримали такі компанії як IBM, Maxtor і Samsung. Hitachi взяла участь у розвитку технології S.M.A.R.T. уже на стадії розробки S.M.A.R.T. II, першими запропонувавши методику повної самодіагностики накопичувача (extended self-test).

У цей час виробники жорстких дисків готуються прийняти до використання новий варіант технології S.M.A.R.T. – "1024 S.M.A.R.T.", характерною рисою якого буде помітно більший розмір журналів, повсюдне використання мультисекторних журналів, більше точні алгоритми аналізу

показань убудованих у накопичувач сенсорів (термодатчики, сенсори ударів, і т.п.) і багато чого іншого. От кілька нових функцій:

– введення алгоритму аналізу температурного режиму накопичувача;

– введення обмеження по мінімальній і максимальній температурі в робочому стані;

– введення лічильника загальної кількості записаних секторів протягом життєвого циклу накопичувача;

– введення лічильника запусків внутрішніх алгоритмів відновлення (recovery counters).

Головним же плюсом можна вважати введення нових атрибутів, які дозволять контролювати стан і робочі характеристики по кожній з головок читання/запису:

– відносна стійкість (стабільність "польоту") головки;

– виправлення помилок читання (з "схованими" повторними спробами);

– автоматичний перерозподіл дефектних ділянок поверхні при операціях запису;

– лічильник-накопичувач G-List для обліку кількості прийнятих ударних навантажень;

– лічильник-накопичувач S-List для обліку загальної кількості "програмних" помилок.

Дані зберігаються в шістнадцатковому виді, називаному «raw value», а потім перераховуються в «value», значення, що символізує надійність щодо деякого еталонного значення. Звичайно «value» розташовується в діапазоні від 0 до 100 (деякі атрибути мають значення від 0 до 200 і від 0 до 253).

Висока оцінка говорить про відсутність змін даного параметра або повільному його погіршенні. Низька говорить про можливий швидкий збій.

Значення, менше, чим мінімальне, при якому виробником гарантується безвідмовна робота накопичувача, означає вихід вузла з ладу.

Програми, що відображають стан S.M.A.R.T.-атрибутів, працюють за наступним алгоритмом:

– Перевіряють наявність підтримки технології S.M.A.R.T. накопичувачем.

– Подають у накопичувач команду запиту S.M.A.R.T.-таблиць.

– Одержують таблиці в буфер додатка.

– Розбирають табличні структури, витягаючи з них номера атрибутів і їхні числові значення.

– Зіставляють стандартизовані номери атрибутів їхнім назвам (іноді – залежно від типу, моделі або фірми-виробника HDD).

– Виводять числові значення в зручному для сприйняття виді.

– Витягають із таблиць прапори атрибутів (ознаки, що характеризують призначення атрибута в рамках конкретної firmware накопичувача, наприклад, «життєво важливий» або «лічильник»).

– На підставі всіх таблиць, значень і прапорів виводять загальний стан пристрою.

Технологія S.M.A.R.T. – Self-Monitoring, Analysis and Reporting Technology (від англ. "Технологія Самодіагностики, Аналізу й Звіту") – була розроблена для підвищення надійності й схоронності даних на жорстких дисках. У більшості випадків, S.M.A.R.T.-сумісні пристрої дозволяють пророчити появу найбільш імовірних помилок і, тим самим, дають користувачеві можливість вчасно зробити резервну копію даних і/або повністю замінити накопичувач до виходу його з ладу.

Методи тестування

Існує два способи запуску тестів S.M.A.R.T.: автономний (off-line) або монопольний (captive). Результат тесту завжди зберігається накопичувачем у даних S.M.A.R.T. При автономному запуску накопичувач повідомляє про успішне завершення команди ДО її ФАКТИЧНОГО виконання й тільки після цього виконує тест. При цьому, по інтерфейсу прапор ЗАЙНЯТИЙ (BSY) не виставляється й накопичувач у будь-який момент готовий приступитися до

виконання чергової інтерфейсної команди, припиняючи роботу тесту. Фактично, тест виконується у фоновому режимі. При запуску тесту в монопольному режимі, по інтерфейсу виставляється прапор ЗАЙНЯТИЙ (BSY) і накопичувач починає безпосереднє виконання тесту в режимі реального часу. Будь-яка інтерфейсна команда під час виконання цього тесту приведе до його переривання й зупинки, після чого накопичувач приступиться до обробки команди, що надійшла.

Різновиди тестів S.M.A.R.T.

Офіційно документовані три види внутрішніх тестів, однак ще існує набір так званих "активних" тестів, функціональні особливості яких різні в різних виробників і для широкої публіки не документовані.

Таблиця 3.1 – Різновиди тестів

№	Назва тесту	off-line	captive
1	Off-line collection	+	-
2	Short Self-test	+	+
3	Extended Self-test	+	+
4	Drive Activity test #1..#4	+	+

Автономне сканування поверхні (off-line read scanning)

Більшість накопичувачів забезпечують підтримку автономного сканування поверхні, що є однією з функцій підпрограми автономного збору даних про стан накопичувача (off-line data collection). При виконанні цієї функції, накопичувач виконує повне сканування поверхні шляхом читання кожного сектора й заміщенням ненадійних секторів на запасні сектори з резервної області (spare area) для запобігання втрати користувальницьких даних.

Якщо під час виконання сканування накопичувач одержує команду по інтерфейсу, то процес сканування переривається й накопичувач приступає до обробки команди, що надійшла. При цьому гарантується максимальний час реагування на команду, що надійшла, – до 2 секунд.

Журнали помилок (S.M.A.R.T. error log)

У більшості сучасних накопичувачів реалізована функція журналювання, помилок або інших подій, що з'являються в плині роботи накопичувача. В основному, накопичувачі надають інформацію про п’ять останіх помилок. При цьому зберігаються останні 5 команд, що надійшли в накопичувач, що передують виникненню цієї помилки, і інша необхідна інформація. Накопичувач може також підтримувати додаткові журнали. Їхня структура, розмір і призначення встановлюються фірмою-виробником. При відновленні мікропрограми накопичувача, всі журнали накопичувача очищаються, а загальна кількість помилок установлюється в значення 0.

Перш ніж перейти до функціональної схеми розглянемо детально як працює технологія S.M.A.R.T.

У журналах зберігається час по внутрішніх годинниках накопичувача, тобто або загальний відпрацьований час на даний момент, або час від моменту останнього включення накопичувача.

Log Directory

Тип: Каталог журналів S.M.A.R.T.

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримка мультисекторних журналів.

Даний журнал являє собою свого роду каталог, у якому зазначені адреси всіх підтримуваних журналів S.M.A.R.T. і їхній розмір у секторах. Максимальна кількість журналів – 255.

Summary Error Log

Тип: Сумарний журнал помилок.

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить інформацію про загальну кількість помилок, зафіксованих накопичувачем з моменту першого включення (або відновлення мікропрограми) і докладні записи про останні 5 помилок. Для кожної з 5 зафіксованих помилок зберігаються останні 5 команд, що надійшли в накопичувач. У цьому журналі зберігаються всі помилки UNC, IDNF, помилки сервосистеми, запису/читання й т.д. При цьому, для кожної команди зберігається значення всіх регістрів, час і поточний стан накопичувача на момент подачі самої команди. Помилки, викликані подачею непідтримуваних команд або командами з помилковими параметрами не фіксуються в журналі. Якщо накопичувач підтримує Comprehensive Error Log, то журнал Summary Error Log дублює останні п'ять записів з журналу Comprehensive Error Log.

Comprehensive Error Log

Тип: Комплексний журнал помилок [S.M.A.R.T. Error Logging].

Вид доступу: тільки читання (RO).

Розмір: 1..51 сектор (максимум 26,112 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить докладну інформацію про загальну кількість помилок, зафіксованих накопичувачем з моменту першого включення (або відновлення мікропрограми) і докладні записи про останні помилки. Максимальна кількість помилок, що зберігаються – 255. Для кожної зафіксованої помилки зберігаються останні 5 команд, що надійшли в накопичувач. У цьому журналі зберігаються всі помилки UNC, IDNF, помилки сервосистеми, запису/читання й т.д. При цьому, для кожної команди зберігається значення всіх регістрів, час і поточний стан накопичувача на момент подачі самої команди. Помилки, викликані подачею непідтримуваних команд або командами з помилковими параметрами не фіксуються в журналі.

Extended Comprehensive Error Log

Тип: Розширений комплексний журнал помилок [S.M.A.R.T. Error Logging].

Вид доступу: тільки читання (RO).

Розмір: 1..65,536 секторів (максимум 32 Мбайт).

Примітка: підтримується 28/ 48-бітна адресація секторів.

Призначення даного журналу аналогічно журналу Comprehensive Error Log і містить у собі копію його записів, однак цей журнал має іншу структуру, що дозволяє реалізувати підтримку як 28-бітної, так і 48-бітної адресації секторів. Максимальна кількість помилок, що зберігаються – 327,680.

Self-test Log

Тип: Журнал результатів самоконтролю [S.M.A.R.T. self-test].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить інформацію про результати виконання команд внутрішньої самодіагностики накопичувача. Журнал може зберігати до 21 запису. При перевищенні цієї кількості, журнал починає заповнюватися заново, перезаписуючи 1-й запис 22-м, 2-й – 23-м і так далі. У кожному записі журналу зберігається регістр із номером тесту, код статусу виконання тесту, час на момент запуску/переривання тесту, номер поточної контрольної точки (або точки останова) тесту, а також LBA-адресу сектора, на якому відбулося переривання/скасування тесту.

Extended Self-test Log

Тип: Розширений журнал результатів самоконтролю [S.M.A.R.T. self-test].

Вид доступу: тільки читання (RO).

Розмір: 1..65,536 секторів (максимум 32 Мбайт).

Примітка: підтримується 28/ 48-бітна адресація секторів.

Призначення даного журналу аналогічно журналу Self-test Log і містить у собі копію його записів, однак цей журнал має іншу структуру, що дозволяє реалізувати підтримку як 28-бітної, так і 48-бітної адресації секторів. Максимальна кількість записів – 1,179,648.

Streaming Performance Log

Тип: Журнал параметрів продуктивності потоків [Streaming].

Вид доступу: тільки читання (RO).

Розмір: 1..65,536 секторів (максимум 32 Мбайт).

Даний журнал містить інформацію про переданий накопичувачу параметрів командами керування режимом Automatic Acoustic Management і Typical Host Interface Sector Time (докладніше – див. ATA/ ATAPI-6 rev 1e). У журналі зберігається набір параметрів, по яких виробляється налаштування накопичувача й переклад у нього в режим, коли всі операції читання/запису можливі тільки спеціальними командами й передача даних відбувається у вигляді безперервного потоку, для якого гарантовані й ураховуються всі часові інтервали (на обробку команди, читання й передачу даних; мінімальні/максимальні затримки, час доступу, позиціонування й т.п.). Докладніше про призначення даного виду журналів можна довідатися з опису технології Audio/Video (AV) Streaming Feature.

Write Stream Error Log

Тип: Журнал помилок потокового запису [Streaming].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується 48-бітна адресація секторів.

Даний журнал містить інформацію про виниклі помилки запису в період роботи накопичувача в потоковому режимі (streaming mode). У цьому журналі зберігається загальна кількість подібних помилок, номер останньої помилки, попереднє і поточне значення регістрів стану й помилки, кількість і LBA-номер сектора, на якому дана помилка була зафіксована. Після читання даного журналу, накопичувач скидає лічильник загальної кількості помилок і очищає журнал. Вміст журналу зберігається тільки під час роботи й очищається в момент наступного включення/вимикання накопичувача або при надходженні сигналу

апаратного скидання (hardware reset). Максимальна кількість помилок, що зберігаються – 31.

Read Stream Error Log

Тип: Журнал помилок потокового читання [Streaming].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується 48-бітна адресація секторів.

Даний журнал містить інформацію про виниклі помилки читання в період роботи накопичувача в потоковому режимі (streaming mode). У цьому журналі зберігається загальна кількість подібних помилок, номер останньої помилки, попереднє і поточне значення регістрів стану й помилки; кількість і LBA-номер сектора, на якому дана помилка була зафіксована. Після читання даного журналу, накопичувач скидає лічильник загальної кількості помилок і очищає журнал. Вміст журналу зберігається тільки під час роботи й очищається в момент наступного включення/вимикання накопичувача або при надходженні сигналу апаратного скидання (hardware reset). Максимальна кількість помилок, що зберігаються – 31.

Delayed LBA Sector Log

Тип: Vendor Specified [General Purpose Logging].

Вид доступу: тільки читання (RO).

Розмір: установлюється виробником (VS).

Примітка: підтримується 48-бітна адресація секторів.

Даний журнал містить LBA-адреси всіх секторів, які були переміщені зі свого нормального фізичного розташування, а також адреси границь недоступної послідовності секторів. У такий спосіб ведеться журнал всіх дефектних або нестабільних секторів. Максимальний розмір журналу встановлюється виробником. Нове фізичне розташування, метод і час доступу до заміщених секторів також установлюється виробником і не документується. Запис у даний журнал може бути додана в будь-який момент часу, за умови активності

(живлення) самого накопичувача. Для процесу відновлення журналу встановлюється найвищий пріоритет і виконання всіх інших команд припиняється. При цьому видалити існуючий запис із журналу не можливо. Вміст журналу зберігається при циклах включення/вимикання накопичувача й при надходженні сигналу апаратного скидання (hardware reset).

ECC Uncorrectable Sector Log

Тип: Журнал непоправних помилок ECC [S.M.A.R.T. Recovering].

Вид доступу: тільки читання (RO).

Розмір: 1 сектор (512 байт).

Примітка: підтримується тільки 28-бітна адресація секторів (28-bit LBA).

Даний журнал містить список LBA-адрес секторів, на яких була зафіксована й зігнорована некоректуєма помилка ECC при виконанні операції READ CONTINUOUS (див. AV feature). При цьому, виконання процедури автоматичного перепризначення збійного сектора (ADR – Automatic Defects Reassigment) накопичувачем заблоковано. Журнал може містити до 126 записів. Даний журнал доступний для читання тільки при дозволеній операції READ CONTINUOUS. У противному випадку накопичувач поверне код помилки ERR->ABRT, перерве виконання команди або поверне порожній журнал. Після успішного читання журналу, у самому накопичувачі він буде очищений.

Reassigned Sector Log

[under construction]

Drive Activity Log

[under construction]

Drive Time Buffer Log

[under construction]

Host Vendor Specific Log

Тип: Користувальницькі журнали.

Вид доступу: читання/запис (R/W).

Розмір: максимум 31 журнал по 16 секторів (253,952 байт).

Примітка: зміст і формат журналу – кожне, на розсуд користувача.

Цей вид журналу може бути використаний для зберігання довільних користувальницьких даних. Для запису цього журналу використовується команда WRITE S.M.A.R.T. LOG. Якщо даний журнал жодного разу не був записаний, то при читанні накопичувач поверне порожній журнал, заповнений нулями.

Device Vendor Specific Log

Тип: Технічні журнали виробника.

Вид доступу: не визначений, на розсуд виробника (VS).

Розмір: максимум 31 журнал по 16 секторів (253,952 байт).

Примітка: зміст, формат і розміри журналу – на розсуд виробника.

Цей вид журналу призначений для внутрішнього використання фірмовими утилітами виробника, для зберігання результатів роботи убудованих підпрограм аналізу й діагностики стану накопичувача й т.п. Можливість читання/запису цього виду журналу встановлюється виробником і не документується. Нові накопичувачі Seagate (моделі Ux і Barracuda ATA) підтримують і навіть реально використовують ще три види журналів S.M.A.R.T., однак їхнє призначення й опис поки не відомі.

Вбудовані функції самоконтролю (self-test)

Практично з моменту появи стандарту S.M.A.R.T. II, у більшості накопичувачів з'явилася нова функція – внутрішня діагностика й самоконтроль, для поглибленого контролю стану механіки накопичувача, поверхні дисків і т.п. Для запуску цієї функції, у набір команд S.M.A.R.T. була введена нова команда – S.M.A.R.T. EXECUTE OFF-LINE IMMEDIATE. Результат роботи зберігається або в спеціалізованих атрибутах, або окремим параметром серед інших даних в атрибутах. Якщо накопичувач підтримує журнали S.M.A.R.T., то результат виконання тестів зберігається також у журналі Self-test Log. Після виконання тесту, накопичувач в обов'язковому порядку обновляє показання у всіх атрибутах і інших параметрах. Якщо під час виконання внутрішнього тесту накопичувач

одержить по інтерфейсу нову команду, то виконання тесту переривається й накопичувач приступає до обробки команди, що надійшла.

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Так, як у схемі є використання атрибутів S.M.A.R.T. розглянемо їх більш детально.

Атрибути

Атрибути S.M.A.R.T. – особливі характеристики, які використовуються при аналізі стану й запасу продуктивності накопичувача. Атрибути вибираються виробником накопичувача, ґрунтуючись на здатності цих атрибутів пророкувати погіршення робочих характеристик накопичувача або визначити його дефектність. Кожний виробник має свій характерний набір атрибутів і може вільно вносити зміни в цей набір у відповідності зі своїми власними вимогами й без повідомлення про це фірм-продавців і кінцевих користувачів.

Значення атрибутів

Значення атрибутів (value) використовуються для подання відносної надійності окремого експлуатаційного або еталонного атрибута. Припустиме значення атрибута лежить у діапазоні від 1 до 255. Високе значення атрибута говорить про те, що результат аналізу даної робочої характеристики вказує на низьку ймовірність її погіршення або виходу накопичувача з ладу. Відповідно, низьке значення атрибута говорить про те, що результат аналізу даної робочої характеристики вказує на високу ймовірність її погіршення або виходу накопичувача з ладу.

Граничні значення атрибутів

Кожний атрибут має власне граничне значення (threshold), що використовується для порівняння зі значенням атрибута (value) і вказує на погіршення робочих характеристик або дефектність накопичувача. Числове значення граничного атрибута визначається виробником накопичувача через конструкційні особливості накопичувача й аналіз результатів випробувань на надійність. Граничне значення кожного атрибута вказує на нижню припустиму границю значення атрибута, аж до якої зберігається позитивний статус

надійності. Граничні значення встановлюються в заводських умовах виробником накопичувача й, у більшості випадків, можуть бути змінені тільки після перемикання накопичувача в технологічний (factory mode). Припустиме граничне значення атрибута може перебуває в діапазоні від 1 до 255.

Якщо значення одного або більше атрибутів, що мають тип pre-failure (в HDD Speed відзначаються символом "*"), менше або дорівнює відповідного граничного значення, то це свідчить про майбутнє погіршення робочих характеристик і/або повному виході накопичувача з ладу.

Короткий опис основних атрибутів

Даний перелік атрибутів є найбільш повним з доступних на сьогоднішній момент у інтернеті або інших джерелах. Призначення атрибутів і спосіб інтерпретації їхніх значень виявлені або досвідченим шляхом, або отримані від служб технічної підтримки компаній-виробників накопичувачів.

Таблиця 3.2 – Значення атрибутів

ID	Назва атрибута
0	= атрибут не використовується
1	Raw Read Error Rate
2	Throughput Performance
3	Spin Up Time
4	Start/Stop Count
5	Reallocated Sector Count
6	Read Channel Margin
7	Seek Error Rate
8	Seek Time Performance
9	Power-On Hours Count
10	Spin Retry Count
11	Recalibration Retries
12	Device Power Cycle Count
13	Soft Read Error Rate
??	Emergency Re-track (Hitachi)

Продовження таблиці 3.2

ID	Назва атрибута
??	ECC On-The-Fly Count (Hitachi)
96	? (Maxtor)
97	? (Maxtor)
98	? (Maxtor)
99	? (Maxtor)
100	? (Maxtor)
101	? (Maxtor)
191	G-Sense Error Rate
192	Power-Off Retract Cycle
193	Load/Unload Cycle Count
194	Temperature
195	? (Quantum AS, Seagate, Maxtor)
196	Reallocation Events Count
197	Current Pending Sector Count
198	Uncorrectable Sector Count
199	UltraDMA CRC Error Rate
200	Write Error Rate (в WD – MultiZone Error Rate)
201	TA Counter Detected
202	TA Counter Increased
203	? (Maxtor)
204	? (Maxtor)
205	? (Maxtor)
206	? (Maxtor)
207	? (Maxtor)
208	? (Maxtor)
209	? (Maxtor)
220	Disk Shift
221	G-Sense Error Rate (в Hitachi – Shock Sense Error Rate)

Продовження таблиці 3.2

ID	Назва атрибута
222	Loaded Hours
223	Load/Unload Retry Count
224	Load Friction
225	Load/Unload Cycle Count
226	Load-in Time
227	Torque Amplification Count
228	Power-Off Retract Count
229	? (IBM DTTA, thanx to Vladislav Shaklein)
230	GMR Head Amplitude
231	Temperature
240	Head Flying Hours (Hitachi)
250	Read Error Retry Rate

Короткий опис відомих атрибутів:

– * – (використовується в програмі HDD Speed) – даний покажчик показує, що відповідний атрибут S.M.A.R.T. є критичним для нормального функціонування накопичувача. Погіршення значень таких атрибутів з найбільшою ймовірністю приводить до виходу накопичувача з ладу. У нових материнських платах BIOS мають убудовану функцію контролю стану накопичувача саме по цих атрибутах.

– Raw Read Error Rate – Частота появи помилок при читанні даних з диска. – Даний параметр показує частоту появи помилок при операціях читання з поверхні диска з вини апаратної частини накопичувача.

– Throughput Performance – Середня продуктивність (пропускна здатність) диска. – Зменшення значення value цього атрибута з великою ймовірністю вказує на проблеми в накопичувачі.

– Spin Up Time – Час розкручування шпинделя. – Середній час розкручування шпинделя диска від 0 RPM до робочої швидкості. Можливо, у поле raw value утримується час у міллісекундах/секундах.

– Start/Stop Count – Кількість циклів запуск/останов шпинделя. – Поле raw value зберігає загальна кількість включень/вимикань диска.

– Reallocated Sectors Count – Кількість перепризначених секторів. – Коли жорсткий диск зустрічає помилку читання/запису/верифікації він намагається перемістити дані з нього в спеціальну резервну область (spare area) і, у випадку успіху, позначає сектор як "перепризначений". Також, цей процес називають remapping, а перепризначений сектор – remap. Завдяки цій можливості, на сучасних жорстких дисках дуже рідко видні [при тестуванні поверхні] так звані bad block. Однак, при великій кількості ремапів, на графіку читання з поверхні будуть помітні "провали" – різке падіння швидкості читання (до 10% і більше). Поле raw value містить загальна кількість перепризначених секторів.

– Read Channel Margin – Запас каналу читання. – Призначення цього атрибута не документовано й у сучасних накопичувачах він не використовується.

– Seek Error Rate – Частота появи помилок позиціонування блоку магнітних головок (БМГ). – У випадку збоїти в механічній системі позиціонування, ушкодження сервометок (servo), сильного термічного розширення дисків і т.п. виникають помилки позиціонування. Чим їх більше, тим гірше стан механіки й/або поверхні жорсткого диска.

– Seek Time Performance – Середня продуктивність операцій позиціонування БМГ. – Даний параметр показує середню швидкість позиціонування привода БМГ на зазначений сектор. Зниження значення цього атрибута говорить про неполадки в механіку привода.

– Power-On Hours – Кількість відпрацьованих годин у включеному стані. – Поле raw value цього атрибута показує кількість годин (мінут, секунд – залежно від виробника), відпрацьованих жорстким диском. Зниження значення (value) атрибута до критичного рівня (threshold) указує на виробіток диском

ресурсу (MTBF – Mean Time Between Failures). На практиці, навіть падіння цього атрибута до нульового значення не завжди вказує на реальне вичерпування ресурсу й накопичувач може продовжувати нормально функціонувати.

– Spin Retry Count – Кількість повторів спроб старту шпинделя диска. – Даний атрибут фіксує загальну кількість спроб розкручування шпинделя і його виходу на робочу швидкість, за умови, що перша спроба була невдалою. Зниження значення цього атрибута говорить про неполадки в механіку привода.

– Recalibration Retries – Кількість повторів спроб рекалібровки накопичувача. – Даний атрибут фіксує загальну кількість спроб скидання стану накопичувача й установки головок на нульову доріжку, за умови, що перша спроба була невдалою. Зниження значення цього атрибута говорить про неполадки в механіку привода.

– Device Power Cycle Count – Кількість повних циклів запуску/останова жорсткого диска.

– Soft Read Error Rate – Частота появи "програмних" помилок при читанні даних з диска. – Даний параметр показує частоту появи помилок при операціях читання з поверхні диска з вини програмного забезпечення, а не апаратної частини накопичувача.

– Emergency Re-track

– ECC On-The-Fly Count

– Load/Unload Cycle Count – Кількість циклів виводу БМГ у спеціальну парковочну зону/у робоче положення. Докладніше – опис технології Head Load/Unload Technology.

– Temperature – Температура. – Даний параметр відбиває в поле raw value показання убудованого температурного сенсора в градусах Цельсія.

– Reallocation Event Count – Кількість операцій перепризначення (ремаппінгу). – Поле raw value цього атрибута показує загальну кількість спроб перепризначення збійних секторів у резервну область, початих накопичувачем. При цьому, ураховуються як успішні, так і невдалі операції.

– Current Pending Sector Count – Поточна кількість нестабільних секторів. – Поле raw value цього атрибута показує загальну кількість секторів, які накопичувач у цей момент уважає претендентами на перепризначення в резервну область (remap). Якщо надалі якийсь із цих секторів буде прочитаний успішно, то він виключається зі списку претендентів. Якщо ж читання сектора буде супроводжуватися помилками, то накопичувач спробує відновити дані й перенести їх у резервну область, а сам сектор позначити як перепризначений (remapped). Постійно ненульове значення raw value цього атрибута говорить про низьку якість (окремої зони) поверхні диска.

– Uncorrectable Sector Count – Кількість нескоректованих помилок. – Атрибут показує загальну кількість помилок, що виникли при читанні/запису сектора і які не вдалося скорегувати. Ріст значення в поле raw value цього атрибута вказує на явні дефекти поверхні й/або проблеми в роботі механіки накопичувача.

– UltraDMA CRC Error Count – Загальна кількість помилок CRC у режимі UltraDMA. – Поле raw value містить кількість помилок, що виникли в режимі передачі даних UltraDMA у контрольній сумі (ICRC – Interface CRC). Практика, зібрана статистика й вивчення журналів помилок S.M.A.R.T. показують: у більшості випадків помилки CRC виникають при сильному завищенні частоти PCI (більше номінальних 33.6 MHz), сильно перекрученому кабелі, а також – з вини драйверів ОС, які не дотримують вимог до передачі/прийому даних у режимах UltraDMA.

– Write Error Rate (Multi Zone Error Rate) – Частота появи помилок при записі даних. – Показує загальну кількість помилок, виявлених під час запису сектора. Чим більше значення в поле raw value (і нижче значення value), тим гірше стан поверхні диска й/або механіки привода.

– Disk Shift – Зрушення пакета дисків щодо осі шпинделя. – Актуальне значення атрибута втримується в поле raw value. Одиниці виміру – не відомі. Подробиці в описі технології G-Force Protection. Зрушення пакета дисків

можливий у результаті сильного ударного навантаження на накопичувач у результаті його падіння або з інших причин.

– G-Sense Error Rate – Частота появи помилок у результаті ударних навантажень. – Даний атрибут зберігає показання ударочуттєвого сенсора – загальна кількість помилок, що виникли в результаті отриманих накопичувачем зовнішніх ударних навантажень (при падінні, неправильній установці, і т.п.). Докладніше в описі технології G-Force Protection.

– Loaded Hours – Навантаження на привод БМГ, викликана загальним наробітком годин накопичувачем. – Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load/Unload Retry Count – Навантаження на привод БМГ, викликана численними повтореннями операцій читання, запису, позиціонування головок і т.п. Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load Friction – Навантаження на привод БМГ, викликане тертям у механічних частинах накопичувача. – Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load/Unload Cycle Count – Загальна кількість циклів навантаження на привод БМГ. – Ураховується тільки період, у плині якого головки перебували в робочому положенні.

– Load-in Time – Загальний час навантаження на привод БМГ. – Можливо, даний атрибут показує загальний час роботи накопичувача під навантаженням, за умови, що головки перебувають у робочому стані (поза парковочною зоною).

– Torque Amplification Count – Кількість зусиль обертаючого моменту привода.

– Power-Off Retract Count – Кількість зафіксованих повторів включення/вимикання живлення накопичувача.

– GMR Head Amplitude – Амплітуда тремтіння головок (GMR-head) у робочому стані.

– Head Flying Hours

– Read Error Retry Rate

Типи атрибутів

Кожний атрибут може мати деякий набір прапорів, що визначають його функціональні особливості. Нижче приводяться всі шість основних типів і їхні короткі описи:

– Pre-failure (PF). Якщо атрибут має цей тип, то поле threshold атрибута містить мінімально припустиме значення атрибута, нижче якого не гарантується працездатність накопичувача й різко збільшується ймовірність його виходу з ладу.

– On-line collection (OC). Указує, що значення даного атрибута обновляється (обчислюється) під час виконання on-line тестів S.M.A.R.T. або ж під час обох видів тестів (on-line/off-line). У противному випадку, значення атрибута обновляється тільки при виконанні off-line тестів.

– Performance related (PR). Указує на те, що значення цього атрибута прямо залежить від продуктивності накопичувача за окремими показниками (seek/throughput/etc. performance). Звичайно обновляється після виконання self-test'ов S.M.A.R.T.

– Error rate (ER). Указує на те, що значення атрибута відбиває відносну частоту помилок по даному параметрі (raw read/write, seek, etc.).

– Events count (EC). Указує на те, що атрибут є лічильником подій.

– Self-preserve (SP). Указує на те, що значення атрибута обновляється й зберігається автоматично (звичайно при кожному старті накопичувача й при виконанні тестів S.M.A.R.T.).

Функціональна схема складається з наступних блоків:

1. Блок читання журналу помилок:

– Журнал параметрів продуктивності потоків.

– Журнал помилок потокового запису.

– Журнал помилок потокового читання.

– Журнал непоправних помилок.

– Користувальницькі журнали.

– Технічні журнали виробника.

– Каталог журналів S.M.A.R.T.

– Сумарний журнал помилок.

– Комплексний журнал помилок.

– Розширений комплексний журнал помилок.

– Журнал результатів самоконтролю.

– Розширений журнал результатів самоконтролю.

2. Блок читання типів атрибутів:

– Error rate (ER). Указує на те, що значення атрибута відбиває відносну частоту помилок по даному параметрі (raw read/write, seek, etc.).

– Events count (EC). Указує на те, що атрибут є лічильником подій.

– Self-preserve (SP). Указує на те, що значення атрибута обновляється й зберігається автоматично (звичайно при кожному старті накопичувача й при виконанні тестів S.M.A.R.T.).

– Pre-failure (PF). Якщо атрибут має цей тип, то поле threshold атрибута містить мінімально припустиме значення атрибута, нижче якого не гарантується працездатність накопичувача й різко збільшується ймовірність його виходу з ладу.

– On-line collection (OC). Указує, що значення даного атрибута обновляється (обчислюється) під час виконання on-line тестів S.M.A.R.T. або ж під час обох видів тестів (on-line/off-line). У противному випадку, значення атрибута обновляється тільки при виконанні off-line тестів.

– Performance related (PR). Указує на те, що значення цього атрибута прямо залежить від продуктивності накопичувача за окремими показниками (seek/throughput/etc. performance). Звичайно обновляється після виконання self-test'ов S.M.A.R.T.

Рисунок 3.2 – Функціональна схема системи

3. Блок автономного сканування поверхні.

4. Блок читання атрибутів:

– Кількість відпрацьованих годин у включеному стані.

– Кількість повторів спроб старту шпинделя диска.

– Кількість повторів спроб рекалібровки накопичувача.

– Кількість повних циклів запуску/останова жорсткого диска.

– Частота появи "програмних" помилок при читанні даних з диска.

– Кількість циклів виводу БМГ у спеціальну парковочну зону/у робоче положення.

– Температура.

– Кількість операцій перепризначення (ремаппінгу).

– Поточна кількість нестабільних секторів.

– Кількість нескоректованих помилок.

– Загальна кількість помилок CRC у режимі UltraDMA.

– Частота появи помилок при записі даних.

– Зрушення пакета дисків щодо осі шпинделя.

– Частота появи помилок у результаті ударних навантажень.

– Навантаження на привод БМГ, викликана загальним наробітком годин накопичувачем.

– Навантаження на привод БМГ, викликана численними повтореннями операцій читання, запису, позиціонування головок і т.п.

– Навантаження на привод БМГ, викликане тертям у механічних частинах накопичувача.

– Загальна кількість циклів навантаження на привод БМГ.

– Загальний час навантаження на привод БМГ.

– Кількість зусиль обертаючого моменту привода.

– Кількість зафіксованих повторів включення/вимикання живлення накопичувача.

– Амплітуда тремтіння головок (GMR-head) у робочому стані.

– Частота появи помилок при читанні даних з диска.

– Середня продуктивність (пропускна здатність) диска.

– Час розкручування шпинделя.

– Кількість циклів запуск/останов шпинделя.

– Кількість перепризначених секторів.

– Запас каналу читання.

– Частота появи помилок позиціонування БМГ.

– Середня продуктивність операцій позиціонування БМГ.

5. Блок вбудованих функцій самоконтролю.

6. Блок вибору методів тесту:

– автономний (off-line);

– монопольний (captive).

7. Блок набору «активних» тестів.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

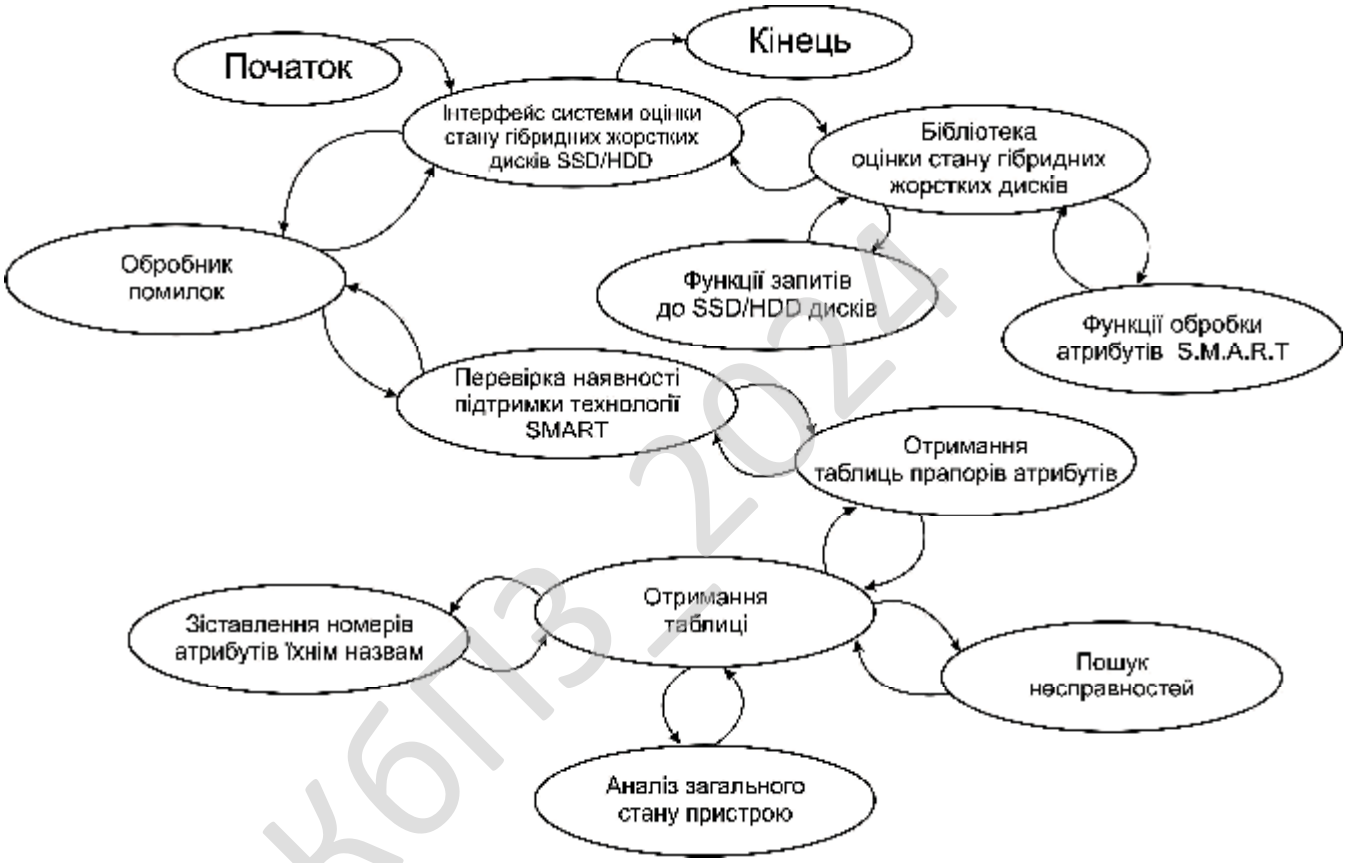


Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над магістерською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю систем оцінки стану гібридних жорстких дисків SSD/HDD.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

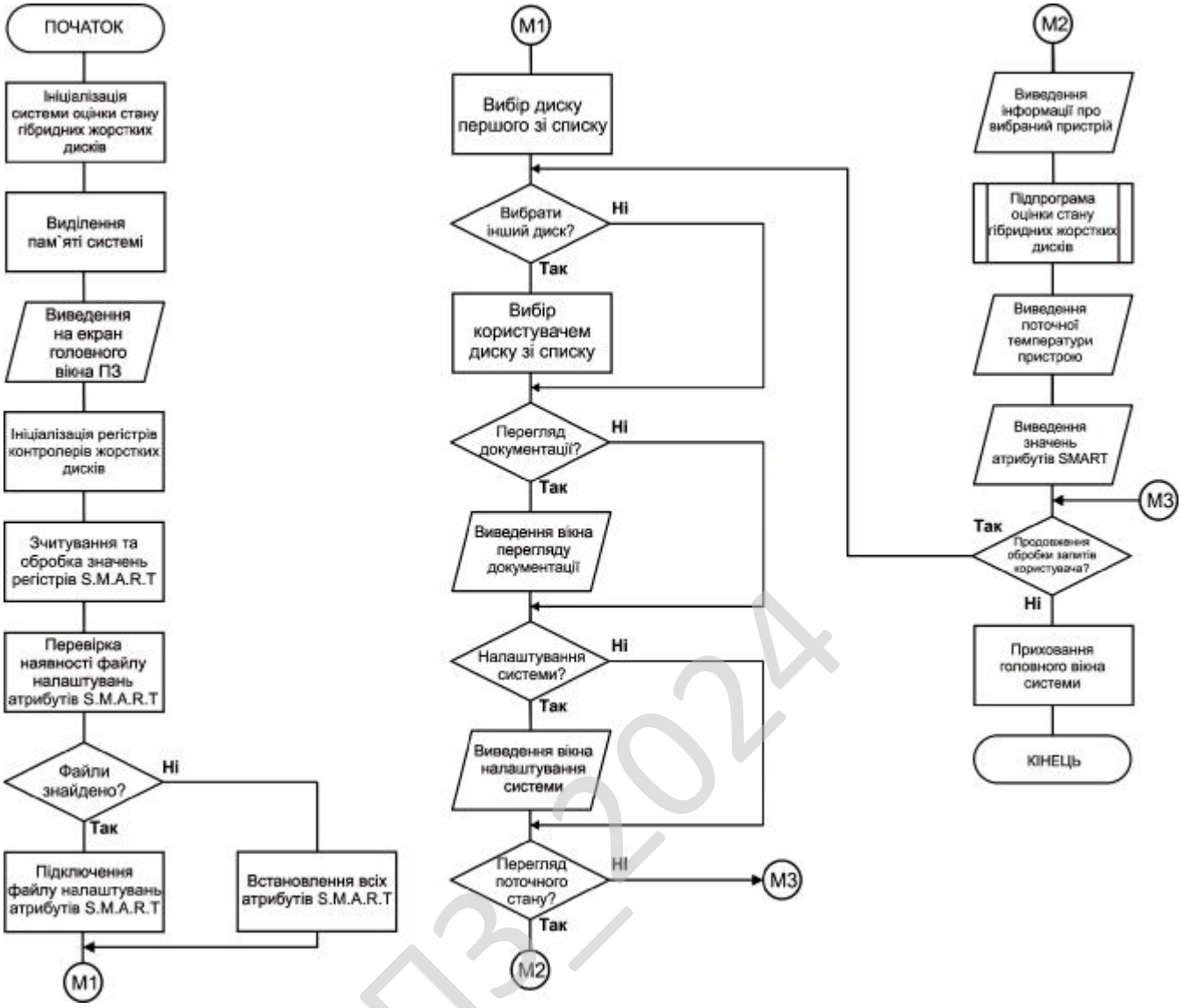


Рисунок 4.1 – Блок-схема основної програми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

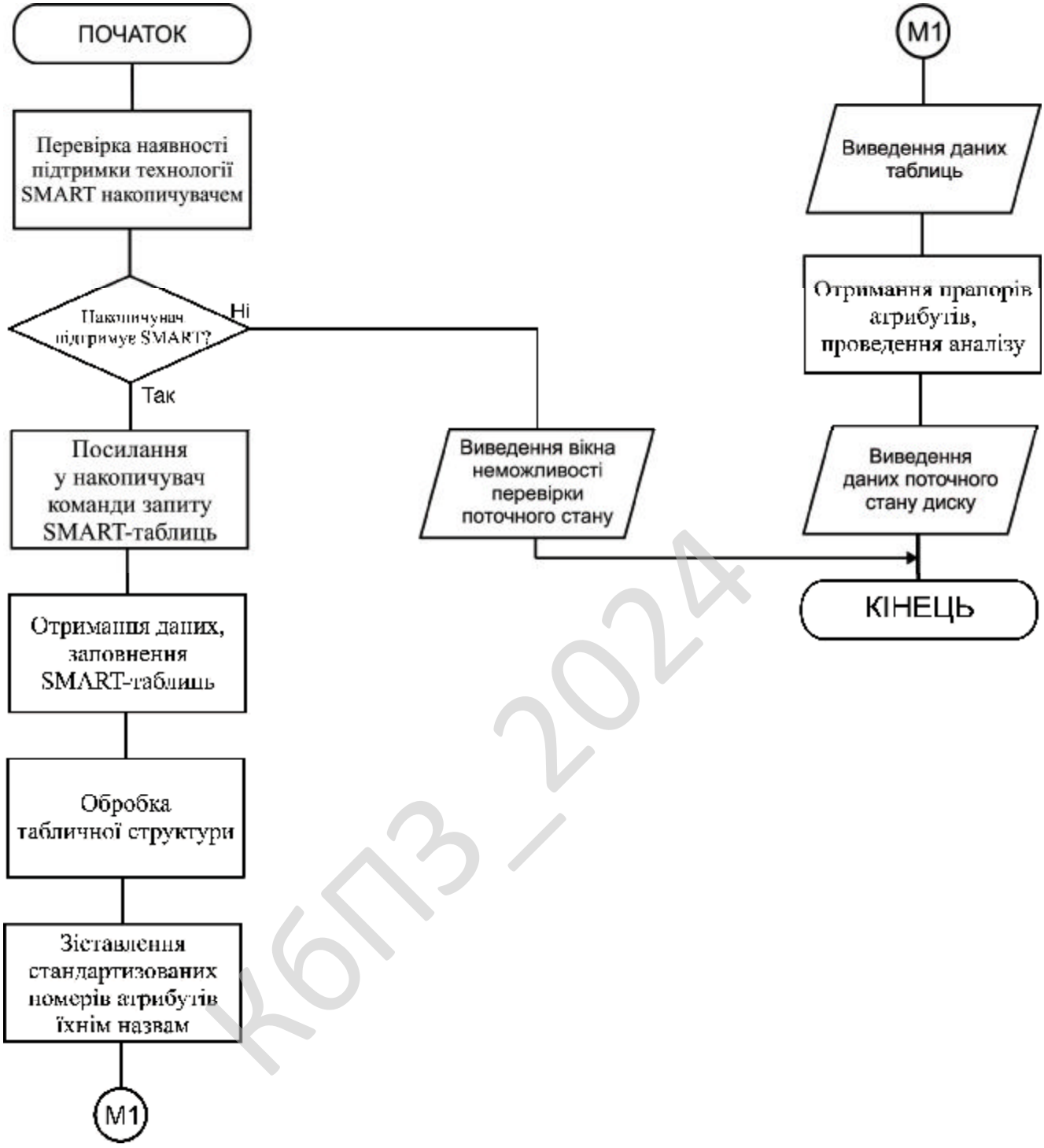


Рисунок 4.2 – Блок-схема роботи підпрограми

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми

автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці магістерської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням

тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі

випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0.. *), "одиниця або більше" (1.. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0.. 1, 3..4, 6.. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з

класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

При розробці я використовував документ «Small Form Factor Committee. Specification for Self-Monitoring, Analysis and Reporting Technology», затверджений такими компаніями, як Compaq Computer Corporation, Hitachi Ltd., IBM Storage Products Company, Maxtor Corporation, Quantum Corporation, Seagate Technology, Toshiba Corporation і Western Digital Corporation. Більшість положень цього документа актуально і по сей день.

Слід також зазначити, що на сьогоднішній день стандарт на технологію SMART не затверджено. Однак у стандарті ATA, починаючи з версії 3, описаний обов'язковий мінімум для технології SMART, і якщо ваш жорсткий диск відповідає ATA (3-8), то він буде підтримувати дану технологію у відповідності з цим стандартом.

Аналіз стану диска проводиться за допомогою вивчення атрибутів SMART Максимальна кількість атрибутів на одному диску залежить від виробника і не перевищує 30. Атрибути можуть приймати значення в діапазоні від 1 до 253.

Для кожного атрибута існує граничне значення, ґрунтуючись на якому можна судити про близькість моменту виходу приводу з ладу.

Розглянемо розроблений код. Для початку необхідно створити дескриптор для роботи з функціями SMART.

Система складається з кількох частин:

1. Моніторинг пристроїв, збір даних про SSD та HDD за допомогою psutil та smartmontools.

2. Аналіз стану дисків, оцінка стану диска за SMART параметрами, таких як температура, кількість помилок та інші.

3. Збереження даних, збір і збереження результатів у JSON форматі для подальшого аналізу.

4. Візуалізація, побудова графіків для оцінки роботи дисків з використанням matplotlib.

```
import psutil
import subprocess
import json
import time
import pandas as pd
import matplotlib.pyplot as plt

# Функція для отримання SMART даних
def get_smart_data(disk):
    try:
        # Викликаємо smartctl для отримання SMART даних
        result = subprocess.check_output(['smartctl', '-a', disk],
encoding='utf-8')
        return result
    except subprocess.CalledProcessError as e:
        # Якщо виникає помилка при виклику smartctl
        print(f"Помилка при зборі даних для {disk}: {e}")
        return None

# Функція для аналізу отриманих SMART даних
def analyze_smart_data(smart_data):
    smart_info = {}
    for line in smart_data.splitlines():
        if "Temperature" in line:
            smart_info['Temperature'] = line.split()[-1]
        if "Reallocated_Sector_Ct" in line:
```

```
                smart_info['Reallocated_Sector_Ct'] = line.split()[-1]
            if "Power_On_Hours" in line:
                smart_info['Power_On_Hours'] = line.split()[-1]
        return smart_info

    # Функція для перевірки стану диска
    def check_disk_status(disk):
        smart_data = get_smart_data(disk)
        if smart_data:
            smart_info = analyze_smart_data(smart_data)
            return smart_info
        else:
            return None

    # Функція для збору інформації про всі диски
    def get_all_disks():
        disks = psutil.disk_partitions()
        disk_list = []
        for disk in disks:
            if 'sd' in disk.device or 'nvme' in disk.device:
                disk_list.append(disk.device)
        return disk_list

    # Функція для збереження результатів у файл
    def save_results_to_json(results):
        with open('disk_status.json', 'w') as f:
            json.dump(results, f, indent=4)

    # Функція для візуалізації стану дисків
    def visualize_results(results):
        df = pd.DataFrame(results)
        df.set_index('Disk', inplace=True)
        df.plot(kind='bar', y='Temperature')
        plt.title('Температура дисків')
        plt.ylabel('Температура (°C)')
        plt.show()

    # Основний цикл моніторингу
    def monitor_disks():
        results = []
        disks = get_all_disks()
```

```
        # Збираємо дані по кожному диску
        for disk in disks:
            disk_info = check_disk_status(disk)
            if disk_info:
                disk_info['Disk'] = disk
                results.append(disk_info)

        # Зберігаємо результати в JSON
        save_results_to_json(results)

        # Візуалізуємо результати
        visualize_results(results)

# Запускаємо моніторинг
while True:
    monitor_disks()
    time.sleep(3600)  # Перевірка кожну годину
```

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 9041:2020. Його повна назва: ДСТУ 9041:2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса.

Цей алгоритм призначений для шифрування коротких (до 616 біт) повідомлень для будь-яких алгоритмів шифрування, в тому числі визначених національними стандартами України.

Як і стандарт цифрового підпису ДСТУ 4145:2002, новий алгоритм використовує криптографічні перетворення у групі точок еліптичних кривих, використовуючи замість кривих у формі Вейєрштрасса найновітніші розробки у галузі еліптичної криптографії – криві у формі Едвардса. Це дає суттєві переваги у швидкодії більш ніж у 3 рази. Новий стандарт розроблений з урахуванням усіх найсучасніших вимог до стійкості криптографічних алгоритмів. Так, нижня межа стійкості криптосистем у цьому стандарті дорівнює 2127 (≈1042) (це більш ніж у

півтора рази вище, ніж у ДСТУ 4145) і можуть бути обрані інші рівні, такі як 2255 (≈1085), 2383 (≈10127) та 2767 (≈10255); крім того, строго обґрунтована його стійкість як до атак на відновлення відкритого тексту, так і до розрізнюючих атак.

Проект алгоритму шифрування, що ліг в основу цього стандарту, пройшов апробацію як в Україні, так і за її межами (Центрально-Європейська конференція з криптографії (червень 2020 року) – форум ведучих криптологів з усього світу).

Стандарт ДСТУ-9041 узгоджений з усіма діючими в Україні національними стандартами. Новиною стандарту є його сфера застосування – інкапсуляція ключів, найсучасніший математичний апарат, а також новий алгоритм генерації псевдовипадкових послідовностей, який, на відміну від аналогічного алгоритму генерації з ДСТУ 4145, використовує виключно національні криптографічні алгоритми національних стандартів та не містить посилань на відповідні пост-радянські стандарти, термін дії яких вже практично вичерпався.

Новий стандарт не належить до так званих постквантових стандартів. Але його стійкість буде під загрозою лише тоді, коли з’являться квантові комп’ютери з 700 і більше кубітами (на даний час кількість "робочих" кубітів, які вдалося створити, – близько 50). Його перевагою перед постквантовими алгоритмами є відносно невелика довжина ключа (у десятки або навіть у сотні разів менша, ніж у постквантових алгоритмах).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання магістерської дипломної роботи. Розроблене програмне забезпечення систем оцінки стану гібридних жорстких дисків SSD/HDD складається з наступних функціональних блоків:

– Блоку відображення атрибутів S.M.A.R.T.

– Блоку функціональних кнопок: Оновити дані; Запуск; Пауза; Налаштування.

– Блоку обрання диску та типу S.M.A.R.T. тестування.

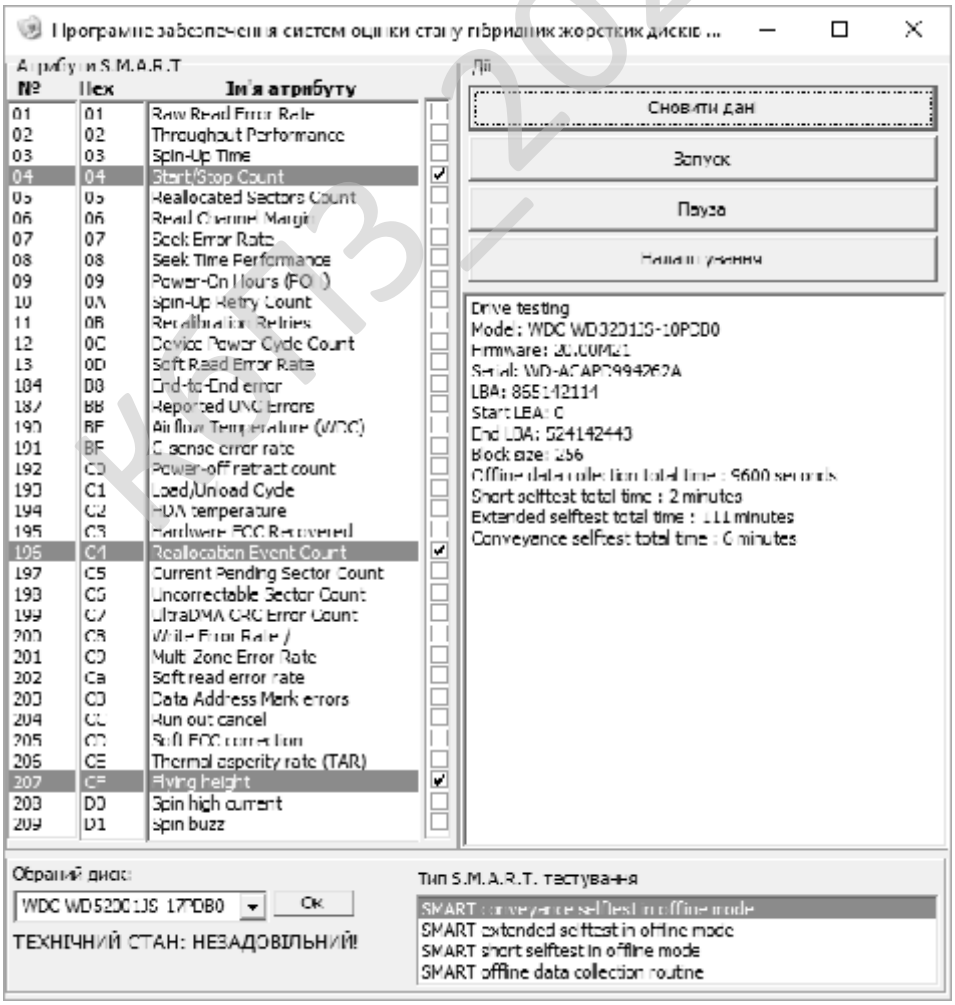


Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

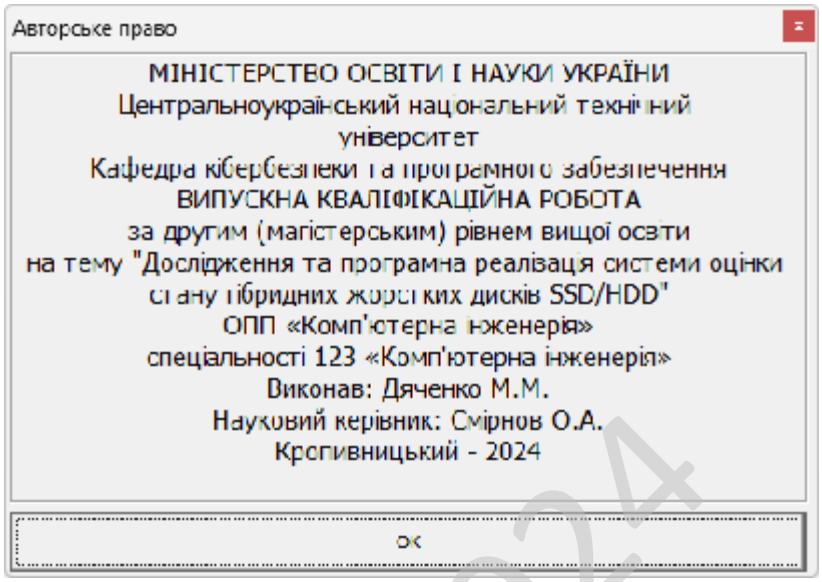


Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи оцінки стану гібридних жорстких дисків SSD/HDD.

Метою розробки є дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD.

Об'єктом дослідження є процес оцінки стану гібридних жорстких дисків SSD/HDD.

Предметом дослідження є методи оцінки стану гібридних жорстких дисків SSD/HDD.

Методи дослідження базуються на методах комп'ютерної інженерії, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод оцінки стану гібридних жорстких дисків SSD/HDD.

– Розроблено вітчизняний продукт оцінки стану гібридних жорстких дисків SSD/HDD, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи оцінки стану гібридних жорстких дисків (SSD/HDD) можуть зацікавити аудиторії, представлені на рисунку 7.1.

ІТ-відділи компаній:	Для забезпечення ефективного моніторингу стану серверів і систем зберігання даних, зниження ризиків втрати інформації.
Провайдери хмарних сервісів:	Для оптимізації роботи дата-центрів та підтримки стабільності інфраструктури зберігання даних.
Виробники обладнання:	Виробники SSD/HDD можуть інтегрувати результати у свої програмні рішення для діагностики та управління дисками.
Центри технічного обслуговування та ремонту:	Для покращення діагностики несправностей та підвищення якості обслуговування клієнтів.
Розробники програмного забезпечення:	Для створення інструментів діагностики та моніторингу стану накопичувачів.
Інженери системного моніторингу:	Для впровадження в системи контролю продуктивності та діагностики комп'ютерного обладнання.
Навчальні заклади та дослідницькі інститути:	Як основа для подальших досліджень у сфері аналізу продуктивності і довговічності носіїв даних.
Користувачі з високими вимогами до зберігання даних:	Наприклад, фрілансери, дизайнери, відеографи та інші професіонали, які працюють із великими обсягами даних.

Рисунок 7.1 – Цільова аудиторія

Результати можуть сприяти зниженню експлуатаційних витрат, підвищенню надійності обладнання та покращенню загальної ефективності роботи систем зберігання даних.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для оцінки привабливості програмної реалізації системи оцінки стану гібридних жорстких дисків SSD/HDD за допомогою методів експертних оцінок можна використовувати метод зважених критеріїв.

1. Визначення критеріїв оцінки (рисунок 7.2)

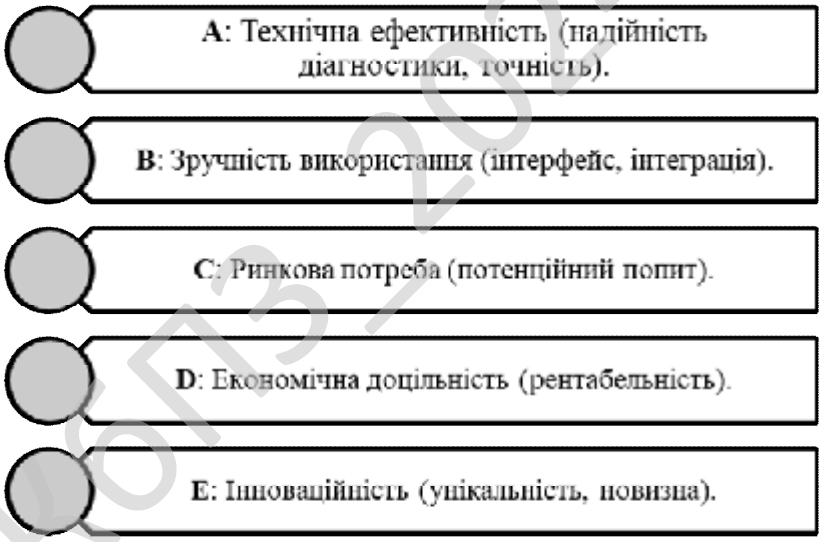


Рисунок 7.2 – Критерії експертної оцінки

2. Залучення 5 експертів: інженери, IT-розробники, представники бізнесу, маркетологи.

3. Встановлення ваг критеріїв. Експерти оцінюють вагу кожного критерію у відсотках (важливість для загальної привабливості): A: 30%, B: 20%, C: 25%, D: 15%, E: 10%.

4. Оцінка за критеріями. Кожен експерт оцінює реалізацію за шкалою від 0 до 10 для кожного критерію і результати зводимо в таблицю 7.1.

Таблиця 7.1 – Оцінки 5-ти експертів за критеріями

Експерт	A (30%)	B (20%)	C (25%)	D (15%)	E (10%)
1	8	7	9	6	8
2	9	8	8	7	9
3	7	9	7	8	8
4	8	7	8	7	7
5	9	8	9	6	8

5. Розрахунок середньої оцінки. Обчислюємо середнє значення оцінок для кожного критерію:

A = (8+9+7+8+9) / 5 = 8.2

B = (7+8+9+7+8) / 5 = 7.8

C = (9+8+7+8+9) / 5 = 8.2

D = (6+7+8+7+6) / 5 = 6.8

E = (8+9+8+7+8) / 5 = 8.0

6. Розрахунок загальної оцінки. Зважуємо середні оцінки за вагами:

Загальна оцінка=A×0.30+B×0.20+C×0.25+D×0.15+E×0.10

Загальнаоцінка=2.46+1.56+2.05+1.02+0.80=7.89

Оцінка 7.89 свідчить про високу привабливість програмної реалізації системи для цільових ринків. Це дає змогу прийняти рішення про подальший розвиток і впровадження проекту. Метод експертних оцінок надає систематичний підхід до оцінки і дозволяє врахувати різні аспекти привабливості.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи оцінки стану гібридних жорстких дисків SSD/HDD найбільш доречними будуть такі методи.

На ранніх етапах розробки: використовуйте витратний підхід для оцінки бюджету та мінімально необхідних інвестицій.

Для оцінки ринкового потенціалу: використовуйте дохідний підхід, якщо є прогнози доходів.

Для обґрунтування ціни перед інвесторами або клієнтами: використовуйте метод ринкових аналогів. Метод ринкових аналогів (Market-Based Method) порівнює вартість подібних рішень на ринку. Ключові етапи: збір інформації про аналогічні системи оцінки стану накопичувачів (вартість розробки або придбання), аналіз ринкових трендів та конкурентів, корекція вартості на основі унікальних характеристик вашої системи (наприклад, підтримка гібридних SSD/HDD).

Для найбільш точного результату можна поєднати кілька методів, а отримані оцінки середньозважити або використовувати як альтернативні сценарії.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Компанія обслуговує дата-центр із 1,000 гібридними накопичувачами SSD/HDD. Раніше діагностика дисків здійснювалася вручну, що займало багато часу і призводило до частих непередбачених простоїв. Нову систему оцінки стану впроваджено для автоматизації моніторингу стану накопичувачів та попередження відмов.

Умовні вхідні дані та результати орієнтовного розрахунку від впровадження системи зводимо в таблицю 7.2.

Таблиця 7.2 – Основні показники впровадження проєкту

1. Основні витрати на впровадження системи Розробка програмного забезпечення**:** $50,000 Ліцензування та апаратне забезпечення: $10,000 Навчання персоналу: $5,000 Експлуатаційні витрати (за рік): $10,000 Загальні витрати: $75,000
2. Економічна вигода 2.1. Зменшення простоїв Середній час простою через відмову одного накопичувача: 5 годин. Вартість простою дата-центру: $2,000/год. Кількість відмов за рік (до впровадження): 50 випадків. Новий прогноз кількості відмов (після впровадження): 20 випадків. Економія: Зменшення простоїв=(50−20)×5×2,000=30×10,000=300,000 USD/рік. 2.2. Зниження витрат на обслуговування Середня вартість ручної діагностики: $200/диск. Кількість діагностик за рік: 1,000. Економія після автоматизації (60% зниження витрат): Економія=1,000×200×0.6=120,000 USD/рік. 2.3. Збільшення терміну експлуатації дисків Середня вартість заміни одного накопичувача: $500. Прогнозована кількість замін до впровадження: 200/рік. Кількість замін після впровадження (завдяки ранній діагностиці): 120/рік. Економія: Економія=(200−120)×500=80×500=40,000 USD/рік.
3. Загальна економія Загальнаекономія=300,000+120,000+40,000=460,000 USD/рік.
4. Розрахунок економічної ефективності Рентабельність інвестицій (ROI): ROI=Загальна економія−Загальні витрати Загальні витрати×100% ROI=460,000−75,000 75,000×100%=513.33%.

Впровадження системи оцінки стану гібридних жорстких дисків SSD/HDD дозволяє знизити витрати на обслуговування, уникнути простоїв і зменшити кількість замін обладнання. Загальна економія становить $460,000 на рік, а рентабельність інвестицій (ROI) перевищує 500%, що робить проект економічно дуже ефективним.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Алгоритм просування проєкту програмної реалізації системи оцінки стану гібридних жорстких дисків SSD/HDD подано на рисунку 7.3.

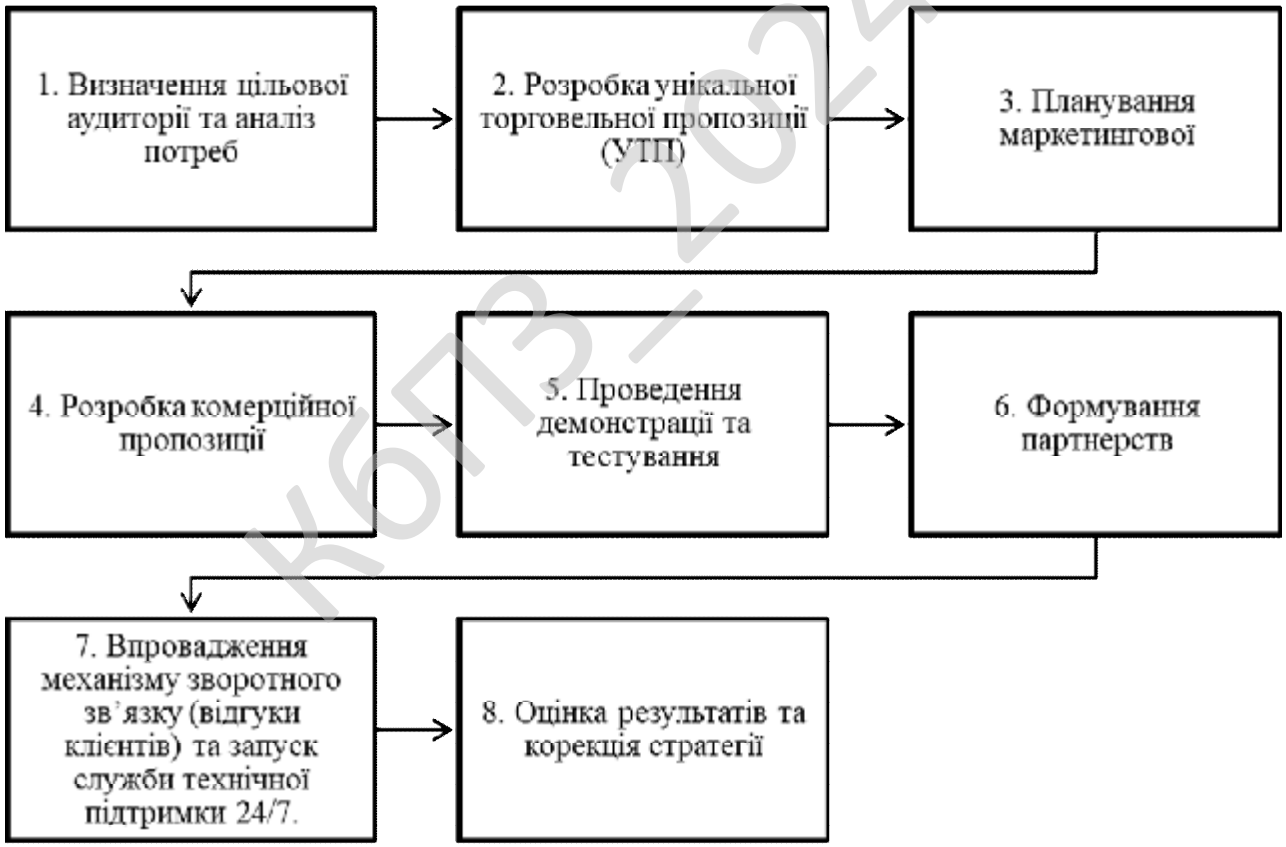


Рисунок 7.3 – Алгоритм просування проєкту

Алгоритм просування дозволяє ефективно презентувати рішення на ринку, привернути увагу цільової аудиторії та досягти економічної окупності проєкту.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Пропозиції щодо оптимізації каналів збуту та шляхів реалізації проєкту програмної реалізації системи оцінки стану гібридних жорстких дисків SSD/HDD має наступний вигляд (рисунок 7.4):

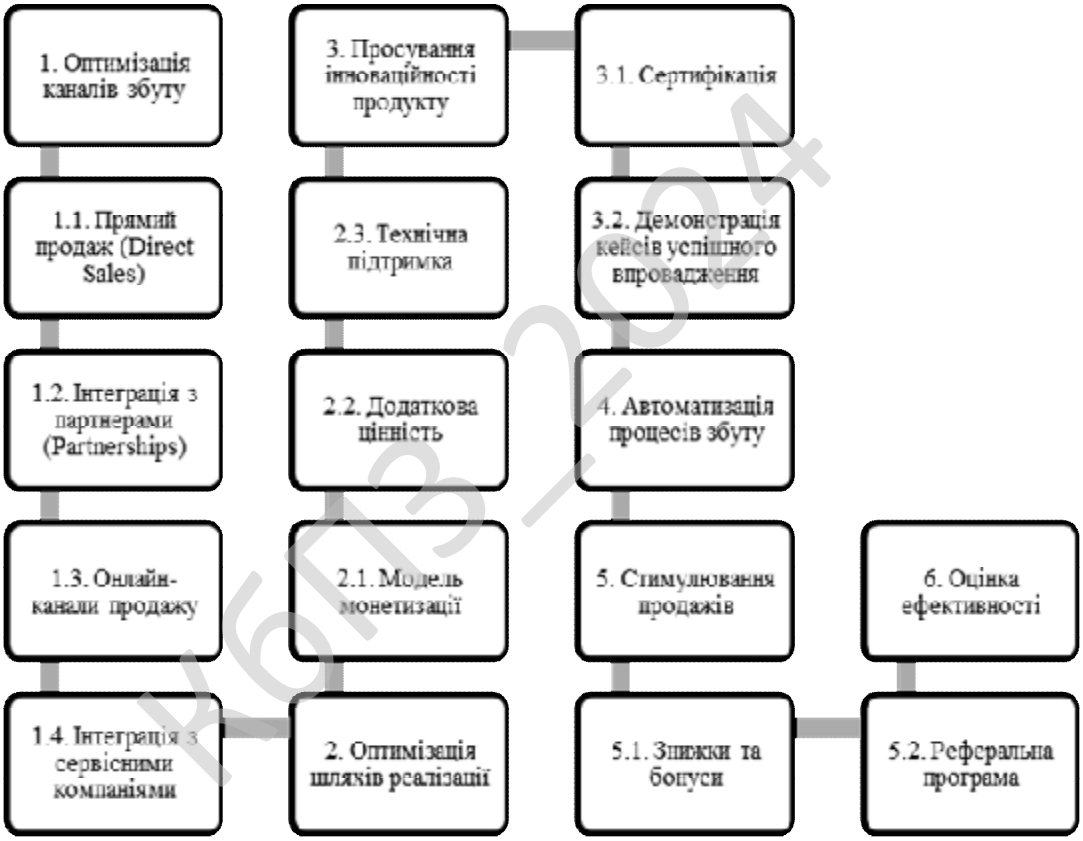


Рисунок 7.4 – Оптимізація каналів збуту та шляхів реалізації

Використання багатоканальної стратегії збуту та різноманітних шляхів реалізації дозволить охопити широку аудиторію, мінімізувати витрати на дистрибуцію та максимізувати прибуток.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключові фактори успіху проєкту програмної реалізації системи оцінки стану гібридних жорстких дисків SSD/HDD схематично подані на рисунку 7.4.

Рисунок 7.5 – Ключові фактори успіху проєкту

Дотримання цих ключових факторів дозволить створити якісний продукт, задовольнити потреби клієнтів і досягти успіху на ринку.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці – це: система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини в процесі трудової діяльності;

Охорона праці є складовою частиною безпеки життєдіяльності [3,4].

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно- обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

Загальний нагляд за додержанням норм охорони праці покладено на прокуратуру, спеціальний покладено на професійні спілки. За безпекою контроль праці здійснюють державні й відомчі спеціалізовані інспекції.

У Законодавстві про працю міститься вимоги і норми з виробничої санітарії, техніки безпеки та норми, що регулюють робочий час, час відпочинку, звільнення та переведення на іншу роботу, а також норми праці щодо жінок, молоді, гігієнічні норми і правила, тощо.

8.2 Аналіз умов праці

Фірма дотримується всіх правил з охорони праці і слідкує за їх дотриманням працівниками.

При виконанні робіт на комп'ютерах працівникам необхідно дотримуватись вимог загальної інструкції з охорони праці.

До роботи на комп'ютерах допускаються особи, які пройшли: медичний огляд, навчання за професією, вступний інструктаж з охорони праці та первинний інструктаж з охорони праці на робочому місці. В подальшому вони проходять

повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки.

Основним обладнанням робочого місця є монітор, системний блок, миша та клавіатура.

Робочі місця розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін – на відстані 1м та між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Монітор розташовується на робочому місці так, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Елементи робочого місця розміщуються так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Джерела освітлення розташовані з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану та клавіатури в напрямку очей користувача від світильників загального освітлення або сонячних променів також використовують антивідблискові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі.

Використовуються скляні поляризаційні фільтри, які забезпечують найкращу якість зображення. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

Зручна робоча поза при роботі з комп'ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Працівники мають пам’ятки, що раціональною робочою позою вважається положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук – вертикальні. Кут ліктєвого суглоба коливається в межах 70 – 90°, зап'ястя зігнуті під кутом не більше ніж 20°, нахил голови 15 – 20°.

Для нейтралізації зарядів статичної електрики в приміщенні, де виконується робота на комп'ютерах, в тому числі на лазерних та світлодіодних

принтерах, збільшується вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити одяг з синтетичних матеріалів.

Для попередження травм усе електричне обладнання заземлене. Приступаючи до роботи, працівникам необхідно перевірити справність обладнання. В разі виявлення порушень їм треба негайно повідомити про це свого керівника для вжиття заходів щодо усунення несправності. Проводити самостійно ремонт електроустаткування забороняється.

8.3 Розробка заходів з охорони праці

Працівники повинні дотримуватися статті 18 Закону України ”Про охорону праці” згідно цій статті працівники зобов'язані:

− знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту;

− співробітничати з власником у справі організації безпечних і нешкідливих умов праці, особисто вживати посильних заходів щодо усунення будь-якої виробничої ситуації, яка створює загрозу його життю чи здоров'ю, або людей, які його оточують, повідомляти про небезпеку свого безпосереднього керівника або іншу посадову особу;

− дотримуватись зобов'язань щодо охорони праці, передбачених колективним договором та правилами внутрішнього трудового розпорядку підприємства.

Також повинні виконуватися вимоги безпеки перед початком роботи, працівникам потрібно:

− увімкнути систему кондиціювання в приміщенні;

− перевірити надійність встановлення апаратури на робочому столі. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній

його край ближче до оператора;

– перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрану;

– відрегулювати освітленість робочого місця;

– відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;

– приєднати до системного блоку необхідну апаратуру. Усі кабелі, що з'єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;

– ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);

– відрегулювати яскравість монітора, мінімальний розмір світної точки, фокусування, контрастність.

Працівники повинні дотримуватися рекомендацій:

– яскравість монітору – не менше 100Kg/м^2;

– відношення яскравості монітора до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;

– мінімальний розмір точки свічення не більше 0,4 мм для монохромного монітора і не менше 0,6 мм для кольорового, контрастність зображення знаку – не менше 0,8.

При виявленні будь-яких неполадок роботу не розпочинати, повідомити про це керівника.

Працівникам потрібно дотримуватися вимог безпеки під час виконання роботи:

– необхідно стійко розташовувати клавіатуру на робочому столі, не опускати її хитання. Під час роботи на клавіатурі сидіти прямо, не напружуватися;

– для уникнення несприятливого впливу на користувача пристроїв типу ”миша” належить забезпечувати вільну велику поверхню столу для переміщення ”миші” і зручного упору ліктьового суглоба;

– не дозволяються сторонні розмови, подразнюючі шуми;

– періодично при вимкненому комп'ютері прибирати ледь змоченою мильним розчином бавовняною ганчіркою порох з поверхонь апаратури. Екран ВДТ та захисний екран протирають ганчіркою, змоченою у спирті. Не дозволяється використовувати рідинні або аерозольні засоби чищення поверхонь комп'ютера.

Працівники не повинні порушувати правил з охорони праці, їм забороняється:

– самостійно ремонтувати апаратуру. Ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера, один раз на півроку повинні відкривати процесор і вилучати пилососом пил і бруд, що накопичилися;

– класти будь-які предмети на апаратуру комп'ютера;

– закривати будь-чим вентиляційні отвори апаратури, що може призвести до її перегрівання і виходу з ладу.

Для зняття статичної електрики рекомендується час від часу доторкатися до металевих поверхонь.

Розташувати принтер необхідно поруч з системним блоком таким чином, щоб з’єднувальний шнур не був натягнутий. Забороняється ставити принтери на системний блок.

Для досягнення найбільш чистих, з високою роздільністю зображень і щоб не зіпсувати апарат, має використовуватися папір, вказаний в інструкції до принтера. При зминанні паперу потрібно відкрити кришку і обережно витягнути лоток з папером.

Працівникам потрібно дотримуватися вимог безпеки після закінчення роботи:

– закінчити та записати у пам'ять комп'ютера файл, що знаходиться в роботі;

– вимкнути принтер та інші периферійні пристрої. Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою для запобігання попаданню в неї пилу;

– прибрати робоче місце;

– ретельно вимити руки теплою водою з милом;

– вимкнути кондиціонер, освітлення і загальне електроживлення;

– пройти в спеціально обладнаному приміщенні сеанс психофізиологічного розвантаження і зняття втоми з виконанням спеціальних вправ аутогенного тренування.

8.4 Пожежна безпека

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів.

З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають правила пожежної безпеки.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймисті рідини, прокладаються тимчасові електропровідники, здійснюється паяння. Виникає додаткова пожежна небезпека, яка потребує відповідних заходів пожежного захисту. До засобів гасіння пожежі,

призначених для локалізації невеликих займань, належать вогнегасники, сухий пісок, азбестові ковдри. Приміщення, в який встановлено комп'ютери і де немає необхідності влаштування систем автоматичного пожежогасіння, необхідно оснащувати переносними вуглекислотними з розрахунку 2 шт. на кожні 20 м^2 в приміщеннях. Звуковбирне облицювання стін, стель приміщень треба виконувати з негорючих та важко горючих матеріалів.

Електроустановки (можливість їх застосування, монтаж, накладка експлуатація) повинні відповідати вимогам чинних правил улаштування електроустановок, правил технічної експлуатації, електроустановок та інших нормативних документів.

Ймовірність виникнення пожежі від електротехнічного та іншого одиничного виробу не повинна перевищувати 10^{-6} на рік. При короткому замиканні в місцях з'єднання проводів опір практично дорівнює нулю, звідси величина струму досягає дуже великих значень.

Персональні комп'ютери після закінчення роботи повинні відключатися від мережі не рідше 1 разу на квартал, необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами. Не дозволяється розміщувати комп'ютерні зали ЕОМ у підвалах; проводити ремонт вузлів (блоків) ЕОМ безпосередньо у залах, де знаходяться ПК (персональні комп'ютери), залишати без нагляду ввімкнену в мережу електронну апаратуру, яка використовується для контролю ЕОМ. Електричний струм силою 0,1 А є небезпечним для людини. Для попередження травм усе електричне обладнання повинне бути заземлене. Приступаючи до роботи необхідно перевірити справність обладнання, ізоляцію проводів і надійність заземлення. Доторкання до оголених струмоведучих і незахищених частин в електроустаткуванні забороняється. В разі виявлення порушень ізоляції електропроводів, відкритих струмоведучих частин електроустаткування або порушення заземлення треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнту використання світлового потоку для приміщення ширина якого складає 2,6 м, довжина – 2,6 м, висота – 3 м.

У зазначеному приміщенні працює 1 особа.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F = E \cdot S \cdot K \cdot Z / n,$$

де: F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S=2,6 х 2,6 = 6,76 $м^2$);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h \cdot (A+B)),$$

де S – площа приміщення, S = 6,76 м2;

h – розрахункова висота підвісу, h = 3 м (співпадає з висотою стелі, оскільки лампи освітлення закріплюються на стелі);

A – ширина приміщення, A = 2,6 м;

B – довжина приміщення, B = 2,6 м.

Підставимо всі значення у формулу та визначимо індекс приміщення: i=0,43.

Знаючи індекс приміщення, знаходимо n = 0,23 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом ламп). Підставимо всі значення у формулу, визначимо світловий потік: F=14548 Лм.

Будемо використовувати світлодіодні панелі Lebron L-LPU-Prismatic, світловий потік яких $F_л$ = 3000 Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де F – світловий потік, $F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначимо індекс приміщення:

$$N= 14548/ 3000=4,8 \text{ шт.}$$

Приймаємо необхідну кількість ламп 5 шт.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров’я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи оцінки стану гібридних жорстких дисків SSD/HDD.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів оцінки стану гібридних жорстких дисків SSD/HDD.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем оцінки стану гібридних жорстких дисків SSD/HDD.

– Досліджена система оцінки стану гібридних жорстких дисків SSD/HDD.

– На основі отриманих результатів досліджень створена програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання оцінки стану гібридних жорстких дисків SSD/HDD.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 9041:2020.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дяченко М.М. Дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

2. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.

3. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.

4. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.

5. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.

6. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

7. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.

8. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

9. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

10. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

11. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

12. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

13. Kuznetsov O., Ilchenko O., Kryvinska N., Buravchenko K., Smirnov O., Savchenko Iu. «An Empirical Assessment of Leading Blockchain Financial Services». *2023 IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF),* Kyiv, Ukraine, 2023, pp. 1-6,

14. Smirnov O., Fedorov E., Neskorodieva A., Neskorodieva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

15. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

16. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems,* 2023, 729 LNNS, pp. 104–112.

17. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

18. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

19.

20. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

21. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

22. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

23. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

24. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

25. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

26. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

27. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

28. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

29. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated

with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

30. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

31. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

32. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

33. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

34. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

35. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

36. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation

Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

37. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

38. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

39. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

40. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

41. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

42. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та

комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

43. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

44. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

45. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

46. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

47. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

48. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

49. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

50. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

51. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи оцінки стану гібридних жорстких дисків SSD/HDD.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи оцінки стану гібридних жорстких дисків SSD/HDD;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати маркетингове та економічне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2024 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута пожежна безпека.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 108 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Маркетингове та економічне обґрунтування ІТ-проєкту.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Смірнов О.А.

Дослідження та програмна реалізація системи оцінки стану гібридних жорстких дисків SSD/HDD

Лістинг програми

Код документу 23

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 23

Літера: РП

Кропивницький – 2024 року

Основна програма

```
import psutil
import subprocess
import json
import time
import pandas as pd
import matplotlib.pyplot as plt
import os

# Функція для перевірки наявності smartctl на системі
def check_smartctl_installed():
    try:
        # Перевіряємо чи доступна команда smartctl
        subprocess.check_output(['smartctl', '--version'], encoding='utf-
8')
        return True
    except FileNotFoundError:
        # Якщо smartctl не знайдено
        print("smartctl не встановлено в системі.")
        return False

# Функція для отримання інформації про диски
def get_smart_data(disk):
    try:
        # Використовуємо команду smartctl для отримання SMART даних
        result = subprocess.check_output(['smartctl', '-a', disk],
encoding='utf-8')
        return result
    except subprocess.CalledProcessError as e:
        # Якщо виникає помилка при виклику smartctl
        print(f"Помилка при зборі даних для {disk}: {e}")
        return None

# Функція для аналізу даних SMART
def analyze_smart_data(smart_data):
    # Словник для зберігання інформації про стан диска
    smart_info = {}

    # Перевірка та збір конкретних значень з SMART даних
    for line in smart_data.splitlines():
        if "Temperature" in line:
            smart_info['Temperature'] = line.split()[-1]
        if "Reallocated_Sector_Ct" in line:
            smart_info['Reallocated_Sector_Ct'] = line.split()[-1]
        if "Power_On_Hours" in line:
```

```
                smart_info['Power_On_Hours'] = line.split()[-1]
            if "Current_Pending_Sector" in line:
                smart_info['Current_Pending_Sector'] = line.split()[-1]
            if "Offline_Uncorrectable" in line:
                smart_info['Offline_Uncorrectable'] = line.split()[-1]

        return smart_info

    # Функція для визначення типу диска (SSD чи HDD)
    def get_disk_type(disk):
        try:
            # Викликаємо команду для отримання інформації про тип диска
            result = subprocess.check_output(['smartctl', '-i', disk],
encoding='utf-8')
            if 'Solid State Device' in result:
                return 'SSD'
            elif 'Hard Disk' in result:
                return 'HDD'
            else:
                return 'Unknown'
        except subprocess.CalledProcessError:
            return 'Unknown'

    # Функція для перевірки стану диска
    def check_disk_status(disk):
        # Отримуємо SMART дані
        smart_data = get_smart_data(disk)
        if smart_data:
            # Аналізуємо отримані SMART дані
            smart_info = analyze_smart_data(smart_data)
            smart_info['Disk'] = disk
            # Додатково визначаємо тип диска (SSD або HDD)
            smart_info['Disk_Type'] = get_disk_type(disk)
            return smart_info
        else:
            return None

    # Функція для отримання всіх дисків у системі
    def get_all_disks():
        # Отримуємо всі підключені диски за допомогою psutil
        disks = psutil.disk_partitions()
        disk_list = []

        for disk in disks:
            # Фільтруємо тільки фізичні диски
            if 'sd' in disk.device or 'nvme' in disk.device:
```

```
            disk_list.append(disk.device)

    return disk_list

# Функція для збереження результатів у файл JSON
def save_results_to_json(results):
    try:
        with open('disk_status.json', 'w') as f:
            json.dump(results, f, indent=4)
    except Exception as e:
        print(f"Помилка при збереженні даних у файл: {e}")

# Функція для візуалізації стану дисків
def visualize_results(results):
    try:
        # Створюємо DataFrame з отриманих результатів
        df = pd.DataFrame(results)
        df.set_index('Disk', inplace=True)
        # Створюємо графік для температури дисків
        df.plot(kind='bar', y='Temperature')
        plt.title('Температура дисків')
        plt.ylabel('Температура (°C)')
        plt.xlabel('Диски')
        plt.show()
    except Exception as e:
        print(f"Помилка при візуалізації даних: {e}")

# Функція для збору даних про стан дисків
def monitor_disks():
    results = []
    disks = get_all_disks()

    # Збираємо дані для кожного диска
    for disk in disks:
        disk_info = check_disk_status(disk)
        if disk_info:
            results.append(disk_info)

    # Зберігаємо отримані результати в JSON файл
    save_results_to_json(results)

    # Візуалізуємо отримані результати
    visualize_results(results)

# Функція для регулярної перевірки стану дисків
def regular_monitoring():
```

```
            while True:
                monitor_disks()
                # Затримка між перевірками
                time.sleep(3600)  # Перевірка кожну годину

        # Функція для виведення зведеної інформації про всі диски
        def summary_report():
            try:
                with open('disk_status.json', 'r') as f:
                    data = json.load(f)

                # Формуємо зведений звіт по стану дисків
                for disk_info in data:
                    print(f"Диск: {disk_info['Disk']}")
                    print(f"Тип диска: {disk_info['Disk_Type']}")
                    print(f"Температура: {disk_info['Temperature']}°C")
                    print(f"Перерозподілені сектори:
{disk_info['Reallocated_Sector_Ct']}")
                    print(f"Час роботи: {disk_info['Power_On_Hours']} годин")
                    print(f"Кількість непідтверджених секторів:
{disk_info.get('Current_Pending_Sector', 'N/A')}")
                    print(f"Кількість непотрібних секторів:
{disk_info.get('Offline_Uncorrectable', 'N/A')}")
                    print('-' * 50)
            except Exception as e:
                print(f"Помилка при формуванні звіту: {e}")

        # Функція для моніторингу здоров'я диска
        def health_check():
            disks = get_all_disks()
            health_status = {}

            for disk in disks:
                disk_info = check_disk_status(disk)
                if disk_info:
                    health_status[disk] = disk_info['Temperature']
                else:
                    health_status[disk] = 'Не доступно'

            print("Здоров'я дисків:")
            for disk, temperature in health_status.items():
                print(f"Диск {disk}: {temperature}°C")

            return health_status

        # Основна функція
```

```
def main():
    if check_smartctl_installed():
        # Запускаємо регулярний моніторинг
        print("Запуск моніторингу стану дисків.")
        regular_monitoring()
    else:
        print("Необхідно встановити smartctl для моніторингу дисків.")
```

Файл check_os.py

```
import platform
import subprocess

# Функція для перевірки операційної системи
def check_os():
    current_os = platform.system()

    if current_os == 'Windows':
        return 'Windows'
    elif current_os == 'Linux':
        return 'Linux'
    elif current_os == 'Darwin':
        return 'macOS'
    else:
        return 'Unknown'

# Функція для отримання SMART даних для кожної операційної системи
def get_smart_data_os_specific(disk):
    current_os = check_os()
    if current_os == 'Windows':
        # Для Windows використовуємо smartmontools через команду wmic або
сторонні утиліти
        try:
            result = subprocess.check_output(['wmic', 'diskdrive', 'get',
'status'], encoding='utf-8')
            return result
        except subprocess.CalledProcessError:
            print("Не вдалося отримати SMART дані для Windows")
            return None
    elif current_os == 'Linux' or current_os == 'macOS':
        # Для Linux та macOS використовуємо smartctl
        try:
            result = subprocess.check_output(['smartctl', '-a', disk],
encoding='utf-8')
            return result
        except subprocess.CalledProcessError:
            print(f"Не вдалося отримати SMART дані для {disk}")
            return None
    else:
        print(f"Невідома операційна система: {current_os}")
        return None
```

Файл ssd_health.py

```
import subprocess
import json
import matplotlib.pyplot as plt

# Функція для отримання SMART даних
def get_smart_data(disk):
    try:
        result = subprocess.check_output(['smartctl', '-a', disk],
encoding='utf-8')
        return result
    except subprocess.CalledProcessError as e:
        print(f"Помилка при зборі SMART даних для {disk}: {e}")
        return None

# Функція для аналізу зносостійкості SSD
def analyze_ssd_health(smart_data):
    health_data = {}
    for line in smart_data.splitlines():
        if "Percentage Used" in line:
            health_data['Percentage Used'] = int(line.split()[-1])
        if "Wear Leveling Count" in line:
            health_data['Wear Leveling Count'] = int(line.split()[-1])
        if "Total NAND Writes" in line:
            health_data['Total NAND Writes (GB)'] = int(line.split()[-2])  #
Значення перед останнім
    return health_data

# Функція для оцінки стану SSD
def evaluate_ssd_health(health_data):
    report = {}
    if health_data['Percentage Used'] >= 80:
        report['Status'] = 'Critical'
        report['Action'] = 'Рекомендується заміна SSD.'
    elif 50 <= health_data['Percentage Used'] < 80:
        report['Status'] = 'Warning'
        report['Action'] = 'Рекомендується моніторинг зносостійкості SSD.'
    else:
        report['Status'] = 'Good'
        report['Action'] = 'Стан SSD в нормі.'
    return report

# Функція для збереження результатів у файл JSON
def save_ssd_health_to_json(disk, health_data, report):
    result = {
        'Disk': disk,
        'Health Data': health_data,
        'Report': report
    }
    with open(f'{disk}_ssd_health.json', 'w') as f:
        json.dump(result, f, indent=4)
    print(f"Результати аналізу збережено в {disk}_ssd_health.json")
```

```
# Функція для візуалізації зносостійкості SSD
def visualize_ssd_health(health_data):
    labels = list(health_data.keys())
    values = list(health_data.values())

    plt.figure(figsize=(10, 6))
    plt.bar(labels, values)
    plt.title("SSD Health Analysis")
    plt.ylabel("Value")
    plt.xlabel("Metric")
    plt.xticks(rotation=45)
    plt.show()

# Основна функція для аналізу стану SSD
def main():
    disk = input("Введіть ім'я SSD диска (наприклад, /dev/sda): ")
    smart_data = get_smart_data(disk)
    if smart_data:
        health_data = analyze_ssd_health(smart_data)
        report = evaluate_ssd_health(health_data)

        print("\nАналіз стану SSD:")
        print(f"Percentage Used: {health_data['Percentage Used']}%")
        print(f"Wear Leveling Count: {health_data['Wear Leveling Count']}")
        print(f"Total NAND Writes: {health_data['Total NAND Writes (GB)']} GB")
        print(f"Status: {report['Status']}")
        print(f"Action: {report['Action']}")

        save_ssd_health_to_json(disk, health_data, report)
        visualize_ssd_health(health_data)
    else:
        print("Не вдалося отримати дані SMART.")

# Запуск програми
if __name__ == "__main__":
    main()
```

Файл database_monitor.py

```
import sqlite3
from datetime import datetime

# Функція для створення бази даних та таблиці
def create_database():
    conn = sqlite3.connect('disk_history.db')
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS disk_status (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            disk TEXT,
            temperature INTEGER,
            reallocated_sectors INTEGER,
            power_on_hours INTEGER,
            timestamp TEXT
        )
    ''')
    conn.commit()
    conn.close()

# Функція для запису даних у базу
def save_to_database(disk, temperature, reallocated_sectors,
power_on_hours):
    conn = sqlite3.connect('disk_history.db')
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO disk_status (disk, temperature, reallocated_sectors,
power_on_hours, timestamp)
        VALUES (?, ?, ?, ?, ?)
    ''', (disk, temperature, reallocated_sectors, power_on_hours,
datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
    conn.commit()
    conn.close()

# Функція для перегляду історії стану дисків
def view_history():
    conn = sqlite3.connect('disk_history.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM disk_status')
    rows = cursor.fetchall()
    conn.close()
    return rows
# Основна функція
def main():
    create_database()
    disk = '/dev/sda'
    save_to_database(disk, 30, 0, 1500)
    save_to_database(disk, 35, 1, 1600)
    history = view_history()
    for entry in history:
        print(entry)
```

Файл notifications_email.py

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# Функція для надсилання сповіщення на email
def send_email_notification(disk, status, action, to_email):
    from_email = 'your_email@example.com'
    password = 'your_password'

    subject = f"Disk Status Alert: {disk}"
    body = f"""
    Disk: {disk}
    Status: {status}
    Action: {action}
    """

    msg = MIMEMultipart()
    msg['From'] = from_email
    msg['To'] = to_email
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))

    try:
        server = smtplib.SMTP('smtp.example.com', 587)
        server.starttls()
        server.login(from_email, password)
        server.sendmail(from_email, to_email, msg.as_string())
        server.quit()
        print(f"Notification sent to {to_email}")
    except Exception as e:
        print(f"Failed to send email: {e}")

# Тестова функція
def main():
    send_email_notification('/dev/sda', 'Critical', 'Replace SSD
immediately', 'recipient@example.com')

# Запуск програми
if __name__ == '__main__':
    main()
```

Файл data_export.py

```
import csv
import xml.etree.ElementTree as ET

# Функція для збереження у формат CSV
def save_to_csv(data, filename):
    with open(filename, 'w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=data[0].keys())
        writer.writeheader()
        writer.writerows(data)

# Функція для збереження у формат XML
def save_to_xml(data, filename):
    root = ET.Element("DiskStatus")
    for entry in data:
        disk_elem = ET.SubElement(root, "Disk")
        for key, value in entry.items():
            elem = ET.SubElement(disk_elem, key)
            elem.text = str(value)
    tree = ET.ElementTree(root)
    tree.write(filename)

# Тестова функція
def main():
    data = [
        {'Disk': '/dev/sda', 'Temperature': 35, 'Reallocated_Sectors': 0,
'Power_On_Hours': 1500},
        {'Disk': '/dev/sdb', 'Temperature': 40, 'Reallocated_Sectors': 1,
'Power_On_Hours': 2000},
    ]
    save_to_csv(data, 'disk_status.csv')
    save_to_xml(data, 'disk_status.xml')
    print("Data saved to CSV and XML")
```

Файл ssd_wear_monitor.py

```
import subprocess
import json
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# Функція для отримання SMART даних
def get_smart_data(disk):
    try:
        result = subprocess.check_output(['smartctl', '-a', disk],
encoding='utf-8')
        return result
    except subprocess.CalledProcessError as e:
        print(f"Помилка при зборі SMART даних для {disk}: {e}")
        return None

# Функція для аналізу зносу SSD
def analyze_ssd_wear(smart_data):
    wear_data = {}
    for line in smart_data.splitlines():
        if "Percentage Used" in line:
            wear_data['Percentage Used'] = int(line.split()[-1])
        if "Wear Leveling Count" in line:
            wear_data['Wear Leveling Count'] = int(line.split()[-1])
        if "Total NAND Writes" in line:
            wear_data['Total NAND Writes (GB)'] = int(line.split()[-2])  #
Передостаннє значення
    return wear_data

# Функція для оцінки стану SSD
def evaluate_ssd_status(wear_data):
    report = {}
    percentage_used = wear_data.get('Percentage Used', 0)
    if percentage_used >= 80:
        report['Status'] = 'Critical'
        report['Action'] = 'Рекомендується негайна заміна SSD.'
    elif 50 <= percentage_used < 80:
        report['Status'] = 'Warning'
        report['Action'] = 'Рекомендується частий моніторинг стану SSD.'
    else:
        report['Status'] = 'Good'
        report['Action'] = 'Стан SSD нормальний.'
    return report

# Функція для надсилання сповіщення на email
def send_email_notification(disk, report, to_email):
    from_email = 'your_email@example.com'
    password = 'your_password'
    subject = f"SSD Status Alert: {disk}"
    body = f"""
    Disk: {disk}
```

```
    Status: {report['Status']}
    Action: {report['Action']}
    """

    msg = MIMEMultipart()
    msg['From'] = from_email
    msg['To'] = to_email
    msg['Subject'] = subject
    msg.attach(MIMEText(body, 'plain'))

    try:
        server = smtplib.SMTP('smtp.example.com', 587)
        server.starttls()
        server.login(from_email, password)
        server.sendmail(from_email, to_email, msg.as_string())
        server.quit()
        print(f"Notification sent to {to_email}")
    except Exception as e:
        print(f"Помилка при відправці email: {e}")

# Функція для збереження результатів у JSON файл
def save_results_to_json(disk, wear_data, report):
    result = {
        'Disk': disk,
        'Wear Data': wear_data,
        'Report': report
    }
    filename = f"{disk.replace('/', '_')}_ssd_wear_report.json"
    with open(filename, 'w') as f:
        json.dump(result, f, indent=4)
    print(f"Результати аналізу збережено в {filename}")

# Основна функція
def main():
    disk = input("Введіть ім'я SSD диска (наприклад, /dev/sda): ")
    email = input("Введіть email для отримання сповіщень: ")

    smart_data = get_smart_data(disk)
    if smart_data:
        wear_data = analyze_ssd_wear(smart_data)
        report = evaluate_ssd_status(wear_data)

        print("\nРезультати аналізу зносу SSD:")
        print(f"Percentage Used: {wear_data.get('Percentage Used', 'N/A')}%")
        print(f"Wear Leveling Count: {wear_data.get('Wear Leveling Count',
'N/A')}")
        print(f"Total NAND Writes: {wear_data.get('Total NAND Writes (GB)',
'N/A')} GB")
        print(f"Status: {report['Status']}")
        print(f"Action: {report['Action']}")

        save_results_to_json(disk, wear_data, report)
```

```
        if report['Status'] in ['Critical', 'Warning']:
            send_email_notification(disk, report, email)
    else:
        print("Не вдалося отримати SMART дані з диска.")
```

Файл collect_disk_stats.py

```
import psutil
import json
import time
from datetime import datetime

# Функція для збору статистики використання диска
def collect_disk_stats():
    disk_stats = psutil.disk_io_counters(perdisk=True)
    stats = {}
    for disk, values in disk_stats.items():
        stats[disk] = {
            'Read Count': values.read_count,
            'Write Count': values.write_count,
            'Read Bytes': values.read_bytes,
            'Write Bytes': values.write_bytes,
            'Read Time (ms)': values.read_time,
            'Write Time (ms)': values.write_time
        }
    return stats

# Функція для збереження статистики у файл JSON
def save_stats_to_json(stats):
    timestamp = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
    filename = f"disk_stats_{timestamp}.json"
    with open(filename, 'w') as f:
        json.dump(stats, f, indent=4)
    print(f"Статистика дисків збережена у файл: {filename}")

# Функція для періодичного збору статистики
def periodic_stat_collection(interval=60):
    while True:
        stats = collect_disk_stats()
        save_stats_to_json(stats)
        time.sleep(interval)

# Запуск програми
if __name__ == '__main__':
    periodic_stat_collection(60)
```

Файл nagios_integration.py

```
import socket

# Функція для створення Nagios плагіну
def nagios_check(disk, status):
    if status == "Good":
        print(f"OK - {disk} is in good condition.")
        exit(0)
    elif status == "Warning":
        print(f"WARNING - {disk} is showing potential issues.")
        exit(1)
    elif status == "Critical":
        print(f"CRITICAL - {disk} needs immediate attention.")
        exit(2)
    else:
        print("UNKNOWN - Unable to determine disk status.")
        exit(3)

# Функція для інтеграції з Zabbix через Trapper
def zabbix_send_metric(disk, metric_name, value,
zabbix_server='127.0.0.1', port=10051):
    message = f"{disk} {metric_name} {value}"
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
        sock.connect((zabbix_server, port))
        sock.sendall(message.encode('utf-8'))
        print(f"Metric sent: {message}")

# Тестова функція
def main():
    disk = '/dev/sda'
    status = 'Critical'
    nagios_check(disk, status)
    zabbix_send_metric(disk, 'disk_status', status)
```

Файл interactive_visualization.py:

```
import pandas as pd
import matplotlib.pyplot as plt

# Функція для створення інтерактивних графіків
def plot_disk_data(data):
    df = pd.DataFrame(data)
    df.set_index('Disk', inplace=True)
    df.plot(kind='bar', figsize=(10, 6))
    plt.title("Disk Status Overview")
    plt.ylabel("Values")
    plt.xlabel("Disks")
    plt.show()

# Тестові дані
def main():
    data = [
        {'Disk': '/dev/sda', 'Temperature': 35, 'Reallocated_Sectors': 0,
'Power_On_Hours': 1500},
        {'Disk': '/dev/sdb', 'Temperature': 40, 'Reallocated_Sectors': 1,
'Power_On_Hours': 2000},
    ]
    plot_disk_data(data)
```

Файл backup_creator.py

```
import shutil
import os
from datetime import datetime

# Функція для створення резервної копії
def create_backup(source_dir, backup_dir):
    if not os.path.exists(backup_dir):
        os.makedirs(backup_dir)
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    backup_path = os.path.join(backup_dir, f"backup_{timestamp}.zip")
    shutil.make_archive(backup_path.replace('.zip', ''), 'zip', source_dir)
    print(f"Backup created: {backup_path}")

# Тестова функція
def main():
    source = '/path/to/source'
    backup = '/path/to/backup'
    create_backup(source, backup)
```

Файл age_analysis.py

```
import datetime

# Функція для аналізу віку диска
def analyze_disk_age(power_on_hours):
    days_in_operation = power_on_hours / 24
    current_age = datetime.timedelta(days=days_in_operation)
    return current_age

# Тестова функція
def main():
    power_on_hours = 15000  # Приклад
    age = analyze_disk_age(power_on_hours)
    print(f"Disk has been in operation for: {age.days} days")
```

Файл disk_benchmark.py

```
import time
import os

# Функція для тестування швидкості запису
def write_speed_test(file_path, file_size):
    data = os.urandom(file_size)
    start_time = time.time()
    with open(file_path, 'wb') as f:
        f.write(data)
    end_time = time.time()
    os.remove(file_path)
    return file_size / (end_time - start_time) / (1024 ** 2)

# Функція для тестування швидкості читання
def read_speed_test(file_path, file_size):
    data = os.urandom(file_size)
    with open(file_path, 'wb') as f:
        f.write(data)
    start_time = time.time()
    with open(file_path, 'rb') as f:
        f.read()
    end_time = time.time()
    os.remove(file_path)
    return file_size / (end_time - start_time) / (1024 ** 2)

# Тестова функція
def main():
    file_path = 'test_file.bin'
    file_size = 100 * 1024 * 1024  # 100 MB
    write_speed = write_speed_test(file_path, file_size)
    print(f"Write Speed: {write_speed:.2f} MB/s")
    read_speed = read_speed_test(file_path, file_size)
    print(f"Read Speed: {read_speed:.2f} MB/s")
```

Файл get_smart_data.py

```
import subprocess
import json
import matplotlib.pyplot as plt
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import psutil
from datetime import datetime

# Конфігурація порогів
THRESHOLDS = {
    "Temperature": 50,
    "Reallocated Sectors": 10,
    "Uncorrectable Errors": 0,
    "Spin Retry Count": 0
}

# Функція для автоматичного виявлення дисків
def detect_disks():
    disks = []
    for disk in psutil.disk_partitions(all=True):
        if disk.device.startswith('/dev'):
            disks.append(disk.device)
    return disks

# Функція для отримання SMART даних
def get_smart_data(disk):
    try:
        result = subprocess.check_output(['smartctl', '-a', disk],
encoding='utf-8')
        return result
    except subprocess.CalledProcessError as e:
        print(f"Помилка при отриманні SMART даних для {disk}: {e}")
        return None

# Функція для парсингу SMART даних
def parse_smart_data(smart_data):
    parsed_data = {}
    for line in smart_data.splitlines():
        if "Reallocated_Sector_Ct" in line:
            parsed_data['Reallocated Sectors'] = int(line.split()[-1])
        if "Power_On_Hours" in line:
            parsed_data['Power On Hours'] = int(line.split()[-1])
        if "Temperature_Celsius" in line:
            parsed_data['Temperature (C)'] = int(line.split()[-1])
        if "Offline_Uncorrectable" in line:
            parsed_data['Uncorrectable Errors'] = int(line.split()[-1])
        if "Spin_Retry_Count" in line:
            parsed_data['Spin Retry Count'] = int(line.split()[-1])
    return parsed_data
```

```
# Функція для оцінки стану HDD
def evaluate_hdd_status(parsed_data):
    status_report = {}
    for metric, threshold in THRESHOLDS.items():
        value = parsed_data.get(metric, 0)
        if value > threshold:
            status_report[metric] = 'Critical'
        else:
            status_report[metric] = 'Good'
    return status_report

# Функція для логування результатів
def log_results(disk, parsed_data, status_report):
    log_entry = {
        'timestamp': datetime.now().strftime('%Y-%m-%d %H:%M:%S'),
        'disk': disk,
        'SMART Data': parsed_data,
        'Status Report': status_report
    }
    with open("disk_monitor_log.json", "a") as log_file:
        log_file.write(json.dumps(log_entry, indent=4) + "\n")
    print("Результати додано до журналу.")

# Функція для генерації HTML звіту
def generate_html_report(disk, parsed_data, status_report):
    html = f"""
    <html>
    <head><title>HDD SMART Report for {disk}</title></head>
    <body>
        <h1>SMART Report for {disk}</h1>
        <table border="1">
            <tr><th>Metric</th><th>Value</th><th>Status</th></tr>
    """
    for key in parsed_data.keys():
        status = status_report.get(key, "N/A")
        html +=
f"<tr><td>{key}</td><td>{parsed_data[key]}</td><td>{status}</td></tr>"
    html += """
        </table>
    </body>
    </html>
    """
    filename = f"{disk.replace('/', '_')}_smart_report.html"
    with open(filename, 'w') as f:
        f.write(html)
    print(f"HTML звіт збережено у файл: {filename}")

# Функція для надсилання звіту електронною поштою
def send_email_report(disk, parsed_data, status_report, recipient):
    html_report = generate_html_report(disk, parsed_data, status_report)
    subject = f"SMART Report for {disk}"
    body = f"""
    <html>
```

```
    <head></head>
    <body>
        <h1>SMART Report for {disk}</h1>
        <p>Attached you will find the SMART report for the disk {disk}.</p>
    </body>
    </html>
    """

    message = MIMEMultipart()
    message['From'] = 'your_email@example.com'
    message['To'] = recipient
    message['Subject'] = subject
    message.attach(MIMEText(body, 'html'))

    # Додавання файлу
    with open(html_report, 'r') as file:
        message.attach(MIMEText(file.read(), 'html'))

    try:
        server = smtplib.SMTP('smtp.example.com', 587)
        server.starttls()
        server.login('your_email@example.com', 'your_password')
        server.send_message(message)
        server.quit()
        print(f"Звіт відправлено на {recipient}")
    except Exception as e:
        print(f"Не вдалося відправити звіт: {e}")

# Функція для візуалізації даних
def visualize_smart_data(parsed_data):
    metrics = list(parsed_data.keys())
    values = list(parsed_data.values())

    plt.figure(figsize=(10, 6))
    plt.bar(metrics, values)
    plt.title("HDD SMART Data Overview")
    plt.ylabel("Values")
    plt.xticks(rotation=45)
    plt.show()

# Основна функція
def main():
    disks = detect_disks()
    if not disks:
        print("Не знайдено жодного доступного диска.")
        return

    for disk in disks:
        print(f"Перевірка диска: {disk}")
        smart_data = get_smart_data(disk)
        if smart_data:
            parsed_data = parse_smart_data(smart_data)
            status_report = evaluate_hdd_status(parsed_data)
```

```
        print(f"\nSMART дані для {disk}:")
        for key, value in parsed_data.items():
            print(f"{key}: {value}")

        log_results(disk, parsed_data, status_report)
        generate_html_report(disk, parsed_data, status_report)
        visualize_smart_data(parsed_data)

        # Надсилання звіту на email, якщо є критичні показники
        if any(status == "Critical" for status in status_report.values()):
            recipient = input("Введіть email для отримання звіту: ")
            send_email_report(disk, parsed_data, status_report, recipient)
    else:
        print(f"Не вдалося отримати SMART дані для {disk}.")
```