ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки
Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Кваліфікаційна робота на тему:
«Методи для моніторингу та оптимізації функціонування ОС»
другого (магістерського) рівня

Виконав:
здобувач гр. СПм-23-5

Шевченко Ілля Олександрович

Керівник роботи:
доц. кафедри ЕОМ

Ткачов Віталій Миколайович

МЕТА РОБОТИ

Метою кваліфікаційної роботи є підвищення ефективності функціонування операційних систем за рахунок впровадження методів моніторингу та оптимізації використання системних ресурсів.

Актуальність роботи полягає в необхідності забезпечення моніторингу та оптимізаційних заходів функціонування ОС на тлі зростання вимог до продуктивності, надійності та інформаційної безпеки, що супроводжується ускладненням архітектури систем, підвищеним навантаженням на ресурси, збільшенням кількості фонових процесів і зростанням ризиків втрати даних.

Методи оптимізації ресурсів ОС

Методи управління:

- Створення та завершення процесів
- Планування
- Синхронізація потоків

Вбудовані інструменти:

- Windows – "Диспетчер завдань"
- macOS – "Монітор системи"
- Linux – команди top, htop, ps

3

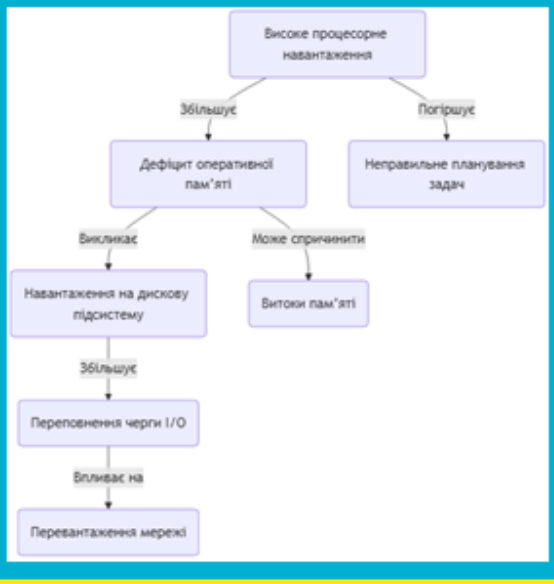
Фактори, що впливають на продуктивність ОС

- Параметри апаратного забезпечення;
- Налаштування та конфігурація системи;
- Встановлене програмне забезпечення;
- Навантаження на процесор, оперативну пам'ять, дискову систему та мережу;
- Споживання енергії та безпека системи.

4

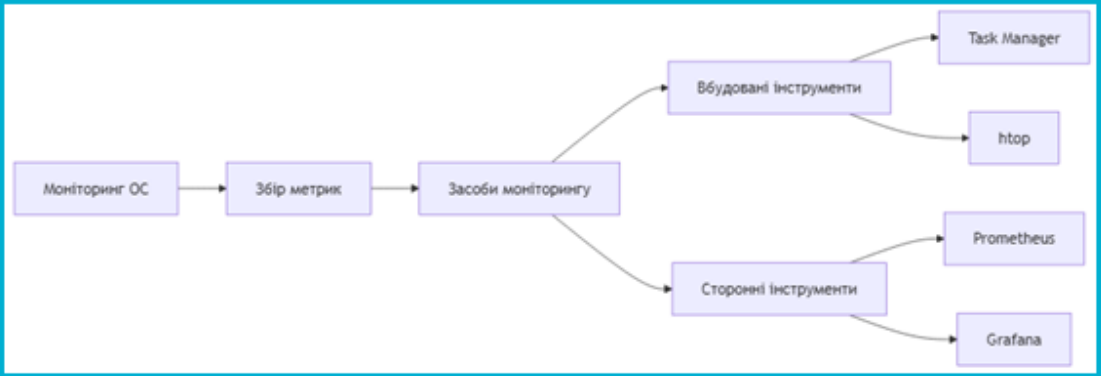
Типові проблеми перевантаження системних ресурсів

- Високе завантаження процесора
- Нестача оперативної пам’яті
- Навантаження на дискову підсистему
- Перевантаження мережі
- Переповнення черги введення/виведення
- Витоки пам'яті
- Неправильне планування задач



5

МЕТОДИ МОНІТОРИНГУ ОС



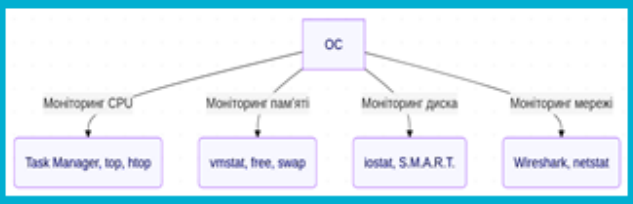
6

МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ UNIX–ПОДІБНИХ СИСТЕМ

Алгоритм запропонованого методу

1. Ініціалізація системного моніторингу.
2. Автоматизоване управління ресурсами.
3. Контроль системного журналювання.
4. Механізм сповіщення та реагування.
5. Циклічний аналіз та адаптація.

Основні компоненти операційної системи, які підлягають моніторингу



7

МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ UNIX–ПОДІБНИХ СИСТЕМ

Ефективність методу

$$\max_t E(t) = w_1 \cdot S(t) - w_2 \cdot R(t) - w_3 \cdot P(t) - w_4 \cdot L(t)$$

Компоненти оптимізаційної задачі

Оцінка стабільності системи

$$S(t) = \frac{T_{uptime}}{T_{total}}$$

Енергоефективність

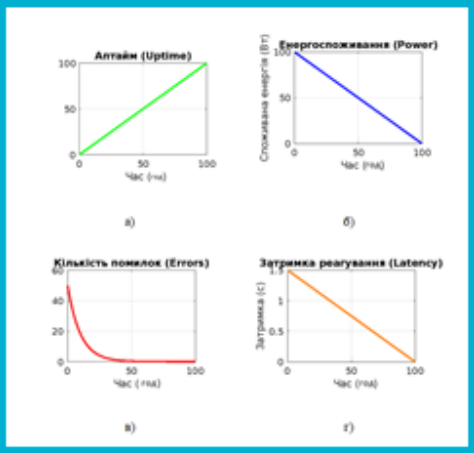
$$P(t) = 1 - \frac{W(t)}{W_{\max}}$$

Оцінка надійності системи

$$R(t) = 1 - \frac{F(t)}{F_{\max}}$$

Швидкість реагування

$$L(t) = 1 - \frac{D(t)}{D_{\max}}$$



Зміни показників ОС Ubuntu 18.04 LTS за часом:
а) час аптайму; б) енергоспоживання ОС;
в) кількість виявлених та виправлених помилок роботи;
г) затримка реагування на події операційною системою

8

Апробація результатів кваліфікаційної роботи

Чепурна, І.С., Шевченко І.О. (2025). Метод підвищення ефективності функціонування Unix–подібних систем. Вісник Херсонського Національного Технічного Університету, №2/2025

ДОВІДКА

Видавничий дім «Гельветика» за домовленістю з Херсонським національним технічним університетом є офіційним видавцем наукового журналу «Вісник Херсонського національного технічного університету» та здійснює усіма видавничо-поліграфічними процесами, до яких належить: набір статей до чергового випуску; редактування; перевірка на плагіат; коректорська вичитка; верстка; присвоєння кожному матеріалу DOI; розміщення електронної версії видання на офіційному сайті журналу; надсилання електронної версії видання до Національної бібліотеки України імені В. І. Вернадського на репозитарис зберігання та представлення на порталі в інформаційному ресурсі «Наукова періодика України»; розсилка обов'язкового безоплатного примірника до наукових установ України.

Цією довідкою повідомляємо, що наукова стаття авторів **Чепурна І. С.** та **Шевченко І. О. «МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ UNIX-ПОДІБНИХ СИСТЕМ»** прийнята редакцією наукового журналу «Вісник Херсонського національного технічного університету» для розміщення у № 2 за 2025 рік.

Директор
Видавничого дому «Гельветика» Олег ГОЛОВКО

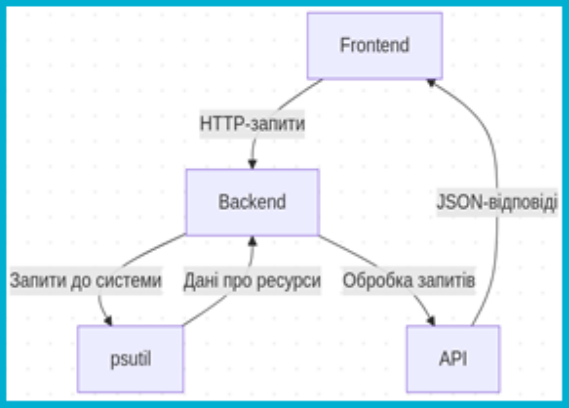
9

ПРОЕКТУВАННЯ ТА РОЗРОБКА ДОДАТКУ ДЛЯ МОНІТОРИНГУ

Архітектура додатку побудована за принципом клієнт-серверної моделі

Додаток складається з трьох основних компонентів:

- Frontend
- Backend
- API



Загальна архітектура застосунку для моніторингу

10

РЕАЛІЗАЦІЯ ЗБОРУ ДАНИХ

Отримання завантаження процесора через stat

```
with open('/proc/stat', 'r') as f:
    cpu_stats = f.readline()
    print("CPU Stats:", cpu_stats)
```

Отримання статистики мережі через dev

```
with open('/proc/net/dev', 'r') as f:
    network_stats = f.readlines()
    for line in network_stats[2:]:
        print("Network Interface Stats:", line.strip())
```

Отримання інформації про оперативну пам'ять

```
import psutil
memory = psutil.virtual_memory()
print("Total Memory:", memory.total)
print("Used Memory:", memory.used)
print("Available Memory:", memory.available)
```

Отримання інформації про диски

```
import psutil
disk_usage = psutil.disk_usage('/')
print("Total Disk Space:", disk_usage.total)
print("Used Disk Space:", disk_usage.used)
print("Free Disk Space:", disk_usage.free)
```

11

Реалізація веб-інтерфейсу

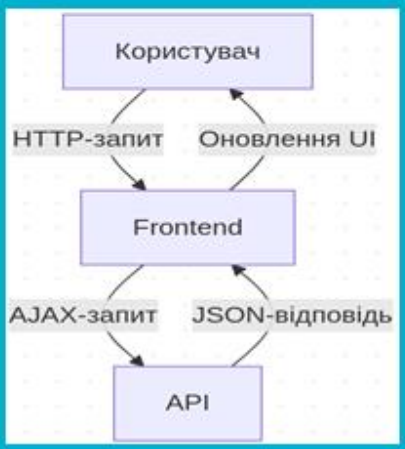


Схема роботи веб-інтерфейсу

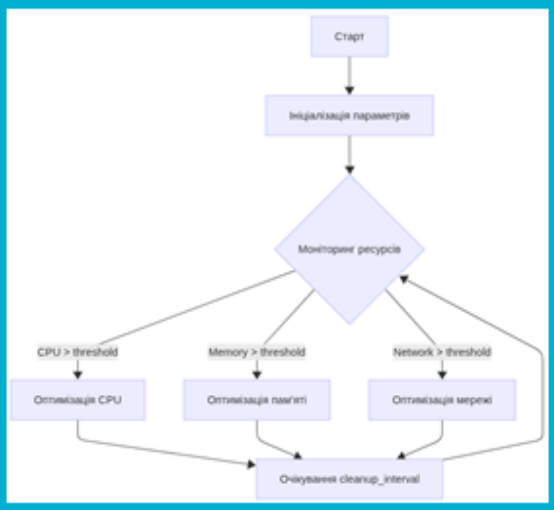
Головне меню програми

12

Автоматизований метод моніторингу та оптимізації функціонування

Перевірка всіх процесів та зупинка неактивних процесів

```
def optimize_cpu(self):
    for proc in psutil.process_iter(['pid', 'name', 'cpu_percent']):
        if proc.info['cpu_percent'] < 1:
            try:
                proc.terminate()
                print(f"Завершено процес {proc.info['name']} (PID: {proc.info['pid']})")
            except psutil.AccessDenied:
                print(f"Відмовлено у доступі для завершення процесу {proc.info['name']}")
```



Логіка роботи системи моніторингу та оптимізації 13

Тестування вебзастосунку для моніторингу ОС

Апаратні конфігурації, використані для тестування вебзастосунку

Тип системи	Операційна система	Процесор	Оперативна пам'ять	Накопичувач
Серверна система	Fedora 41	8-ядерний x86_64	16 ГБ	SSD
Одноплатний комп'ютер	Raspberry Pi OS / Linux	4-ядерний ARM (Raspberry Pi 4)	4 ГБ	microSD
Віртуальна машина	Linux	2 ядра (x86_64)	2 ГБ	Віртуальний диск

Споживання системних ресурсів додатком під час тестування

Система	Середнє навантаження на CPU	Середнє споживання RAM	Примітка
Fedora 41	1–2 %	≈50 МБ	Помірне навантаження, характерне для серверних систем
Raspberry Pi 4	3–5 %	≈60 МБ	Прийнятне для обмежених за ресурсами пристроїв
Віртуальна машина	2–3 %	≈55 МБ	Низьке навантаження при роботі в обмеженому середовищі

14

Тестування вебзастосунку для моніторингу ОС

Результати тестування ефективності бекенду при різному навантаженні

Кількість запитів	Середній час відповіді, мс	CPU, %	Пам'ять, %
600	15.2	3.1	30.5
3000	18.7	7.5	32.1
15000	27.4	18.9	35.8

Результати тестування ендпоінтів

Ендпоінт	Низьке навантаження, мс	Середнє навантаження, мс	Високе навантаження, мс
/api/status	16.1	19.3	29.2
/api/processes	14.8	17.5	25.7
/api/backend_efficiency	2.3	2.7	3.5

15

ВИСНОВОК

- В кваліфікаційній роботі здійснено всебічний аналіз сучасних методів моніторингу та оптимізації функціонування операційних систем.
- Розроблено метод підвищення ефективності функціонування операційної системи в умовах інтенсивного навантаження та обмежених апаратних ресурсів. Запропонований підхід базується на автоматизації управління ключовими системними ресурсами та впровадженні оптимізаційних заходів, що реалізуються на основі результатів моніторингу та діагностики за допомогою вбудованих утиліт операційної системи.
- Результати проведеного аналізу підтвердили, що запропонований метод дозволяє своєчасно виявляти та локалізувати потенційні збої у функціонуванні системи, мінімізувати ризики надмірного використання апаратних ресурсів та забезпечити стабільність та безперервність роботи серверної інфраструктури в умовах підвищеного навантаження.
- Отримані результати можуть бути використані для побудови адаптивних систем моніторингу та оптимізації функціонування серверних операційних систем в корпоративному середовищі.

16

ДОДАТОК Б

Довідка про прийняття статті

«Метод підвищення ефективності функціонування UNIX-подібних систем»

до публікації в журналі «Вісник Херсонського національного технічного університету»

ГЕЛЬВЕТИКА
ВИДАВНИЧИЙ ДІМ

вул. Інглезі, 6/1,
м. Одеса, Україна, 65101
www.helvetica.ua
mailbox@helvetica.ua

Стаціонар: 048 709 38 69
Vodafone: 095 934 48 28
Kyivstar: 097 723 06 08

ДОВІДКА

Видавничий дім «Гельветика» за домовленістю з Херсонським національним технічним університетом є офіційним видавцем наукового журналу «Вісник Херсонського національного технічного університету» та займається усіма видавничо-поліграфічними процесами, до яких належить: набір статей до чергового випуску; рецензування; перевірка на плагіат; коректорська вичитка; верстка; присвоєння кожному матеріалу DOI; розміщення електронної версії видання на офіційному сайті журналу; надсилання електронної версії видання до Національної бібліотеки України імені В. І. Вернадського на репозитарне зберігання та представлення на порталі в інформаційному ресурсі «Наукова періодика України»; розсилка обов'язкового безоплатного примірника до наукових установ України.

Цією довідкою повідомляємо, що наукова стаття авторів **Чепурна І. С.** та **Шевченко І. О. «МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ UNIX–ПОДІБНИХ СИСТЕМ»** прийнята редакцією наукового журналу «Вісник Херсонського національного технічного університету» для розміщення у № 2 за 2025 рік.

Директор
Видавничого дому «Гельветика» Олег ГОЛОВКО

ДОДАТОК В

Лістинг псевдокоду запропонованого методу

```
1. START MonitoringSystem
2. INIT monitoring_scripts
   EVERY 5 min:
       RUN `free`, `vmstat`, `uptime`
       SAVE to /var/log/perf_metrics.log
3. INIT crontab_tasks
   DAILY:
       DELETE files in /tmp, /var/tmp
       RUN `df -h`, `du -sh /var/*`
       IF disk_usage > threshold THEN
           REMOVE oldest files from /var/log/
       ENDIF
       CHECK services [sshd]
       IF service_down THEN
           RESTART service
       ENDIF
4. INIT log_analyzer
   EVERY hour:
       SCAN dmesg for errors
       SCAN /var/log/syslog, auth.log, kern.log
           FOR keywords [fail, error, panic, unauthorized]
           IF found THEN
               LOG to security_alerts.log
               SEND alert to admin (email/Telegram)
           ENDIF
5. INIT notification_dispatcher
   IF alert_generated THEN
       FORMAT message with timestamp + error
       SEND to:
           - Email address
           - Telegram Bot
           - Monitoring System API
       ENDIF
6. INIT weekly_analysis
   EVERY week:
       PARSE /var/log/perf_metrics.log
       BUILD baseline using mean, stddev
       IF deviation > threshold THEN
           APPLY:
               - renice high-CPU tasks
               - ionice disk-heavy processes
               - background non-critical daemons
       ENDIF
7. END MonitoringSystem
```

ДОДАТОК Г

Лістинг коду файлу main.py застосунку для моніторингу ОС

```
import traceback
from aiohttp import web
import ssl
import os
import asyncio
import time

from lib.config import config
from lib.machine import Machine
from lib.cache import Cache
from lib.optimizer import Optimizer
from lib.backend_efficiency import backend_efficiency


machine = Machine()
cache = Cache()

working_dir = os.path.dirname(os.path.realpath(__file__))


async def get_status():
     if cache.should_update():
          info = await machine.get_full_info()
          cache.update(info)
     return cache.get()


routes = web.RouteTableDef()


@routes.get("/")
async def index(request):
     return web.FileResponse("html/index.html")


@routes.get("/api/status")
async def api(request):
     try:
          return web.json_response(await get_status())
     except:
          report =
traceback.format_exc().replace(f"{working_dir}/", "")
          return web.Response(text=report, status=500)
```

```
@routes.get("/api/processes")  # Add a new endpoint for
processes
async def api_processes(request):
     try:
          return
web.json_response(machine.processes.get_processes())
     except:
          report =
traceback.format_exc().replace(f"{working_dir}/", "")
          return web.Response(text=report, status=500)


@routes.get("/api/backend_efficiency")
async def api_backend_efficiency(request):
    return web.json_response(backend_efficiency.get_metrics())


@web.middleware
async def redirector(request, handler):
     try:
          resp = await handler(request)
          if config.get("server", "enable_cors"):
               resp.headers["Access-Control-Allow-Origin"] = "*"
          return resp

     except (web.HTTPInternalServerError, web.HTTPForbidden,
web.HTTPNotFound):
          raise web.HTTPFound(location="/")


@web.middleware
async def efficiency_middleware(request, handler):
     start = time.time()
     response = await handler(request)
     elapsed = time.time() - start
     backend_efficiency.record_request(elapsed)
     return response


async def start_optimizer(app):
    optimizer = Optimizer()
    app['optimizer_task'] =
asyncio.create_task(optimizer.monitor_and_optimize())


routes.static("/", "html")
app = web.Application(middlewares=[redirector,
efficiency_middleware])
app.logger.manager.disable = 100 * config.get("misc", "debug")
app.add_routes(routes)
app.on_startup.append(start_optimizer)
```

```
ssl_context = None

if config.get("server", "domain"):
     ssl_context =
ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
     ssl_dir = f"/etc/letsencrypt/live/{config.get('server',
'domain')}"

     pubkey = config.get("server", "tls_cert_path")
     if not pubkey:
          pubkey = f"{ssl_dir}/fullchain.pem"

     privkey = config.get("server", "tls_key_path")
     if not privkey:
          privkey = f"{ssl_dir}/privkey.pem"

     ssl_context.load_cert_chain(pubkey, privkey)


def main():
    web.run_app(
        app,
        host=config.get("server", "address"),
        port=int(config.get("server", "port")),
        ssl_context=ssl_context
    )


if __name__ == "__main__":
    main()


import os
import psutil  # Потрібно додати до requirements.txt
import asyncio

class Optimizer:
    def __init__(self):
        self.cpu_threshold = 50  # Поріг завантаження CPU (%)
        self.memory_threshold = 50  # Поріг використання пам'яті
(%)
        self.network_threshold = 100 * 1024 * 1024  # Поріг
мережевого трафіку (байт/сек)
        self.cleanup_interval = 60  # Інтервал очищення (секунд)

    async def monitor_and_optimize(self):
        while True:
            await self.check_cpu()
            await self.check_memory()
            await self.check_network()
            await asyncio.sleep(self.cleanup_interval)
```

```
    async def check_cpu(self):
        cpu_usage = psutil.cpu_percent(interval=1)
        if cpu_usage > self.cpu_threshold:
            print(f"High CPU usage detected: {cpu_usage}%.
Taking action...")
            self.optimize_cpu()

    async def check_memory(self):
        memory = psutil.virtual_memory()
        if memory.percent > self.memory_threshold:
            print(f"High memory usage detected:
{memory.percent}%. Taking action...")
            self.optimize_memory()

    async def check_network(self):
        net_io = psutil.net_io_counters()
        if net_io.bytes_sent + net_io.bytes_recv >
self.network_threshold:
            print(f"High network traffic detected. Taking
action...")
            self.optimize_network()

    def optimize_cpu(self):
        # Припинення неактивних процесів
        for proc in psutil.process_iter(['pid', 'name',
'cpu_percent']):
            if proc.info['cpu_percent'] < 1:
                try:
                    proc.terminate()
                    print(f"Terminated process
{proc.info['name']} (PID: {proc.info['pid']})")
                except psutil.AccessDenied:
                    print(f"Access denied to terminate process
{proc.info['name']}")

    def optimize_memory(self):
        # Очищення кешу
        os.system("sync; echo 3 > /proc/sys/vm/drop_caches")
        print("Memory cache cleared.")

    def optimize_network(self):
        # Дії для оптимізації мережі (наприклад, обмеження
швидкості)
        print("Network optimization not implemented yet.")

# Для запуску оптимізатора
async def main():
    optimizer = Optimizer()
    await optimizer.monitor_and_optimize()

if __name__ == "__main__":
    asyncio.run(main())
```

```
from time import time


class Cache:
     def __init__(self, seconds: int=1):
          self.cache = None
          self.seconds = seconds
          self.updated = 0


     def get(self):
          return self.cache


     def should_update(self):
          t = time()
          diff = t - self.updated
          if diff >= self.seconds:
               self.updated = t
               return True
          return False


     def update(self, cache):
          self.cache = cache
```

```
import time
import threading
import psutil

class BackendEfficiency:
    def __init__(self):
        self.request_count = 0
        self.total_response_time = 0.0
        self.lock = threading.Lock()
        self.start_time = time.time()

    def record_request(self, response_time):
        with self.lock:
            self.request_count += 1
            self.total_response_time += response_time

    def get_metrics(self):
        uptime = time.time() - self.start_time
        avg_response = (self.total_response_time /
self.request_count) if self.request_count else 0
        cpu = psutil.cpu_percent()
        mem = psutil.virtual_memory().percent
        return {
            "uptime_sec": uptime,
            "requests": self.request_count,
```

```
            "avg_response_time_ms": avg_response * 1000,
            "cpu_percent": cpu,
            "memory_percent": mem
        }

backend_efficiency = BackendEfficiency()
```