УДК 004.45/.4`6 DOI https://doi.org/10.35546/kntu2078-4481.2025.2.2.46

І. С. ЧЕПУРНА
асистентка кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0008-2442-6221

І. О. ШЕВЧЕНКО
магістрант кафедри електронних обчислювальних машин
Харківський національний університет радіоелектроніки
ORCID: 0009-0006-1867-374X

МЕТОД ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ФУНКЦІОНУВАННЯ UNIX-ПОДІБНИХ СИСТЕМ

В статті пропонується метод підвищення ефективності функціонування UNIX-подібних операційних систем, спрямований на забезпечення стабільної та надійної роботи серверної інфраструктури корпоративної мережі в умовах високого навантаження та обмежених апаратних ресурсів. Цей підхід передбачає використання механізмів автоматизації основних системних ресурсів, що сприяють безперервній роботі серверної інфраструктури, а також реалізацію оптимізаційних заходів на основі методів моніторингу та діагностики за допомогою вбудованих системних утиліт. Запропонований метод дозволяє своєчасно виявляти та усувати потенційні збої, запобігати надмірному використанню ресурсів та забезпечувати стабільну та безперервну роботу системи в умовах високого навантаження. В роботі надаються рекомендації щодо автоматизації процесів моніторингу та управління ресурсами, а також можливості інтеграції запропонованого методу з централізованими системами управління, що використовують засоби раннього виявлення в режимі реального часу. Використання інтелектуальних підходів до адміністрування сприяє підвищенню загальної продуктивності, стабільності та енергоефективності серверної інфраструктури. Особлива увага приділяється оптимізаційним заходам, що зменшують навантаження на апаратні ресурси, зменшують затримки при обробці великих даних та підвищують енергоефективність серверної інфраструктури. Ефективність реалізації запропонованого методу було перевірено шляхом моделювання перевантаження серверної системи локальної комп'ютерної мережі кафедри електронних обчислювальних машин Харківського національного університету радіоелектроніки. Результати підтвердили доцільність та ефективність застосування запропонованого підходу для забезпечення стабільної роботи серверів в сучасних корпоративних мережах.

Ключові слова: *метод, автоматизація, моніторинг, надійність, сервер.*

I. S. CHEPURNA
Assistant at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0008-2442-6221

I. O. SHEVCHENKO
Master's Student at the Department of Electronic Computing Machines
Kharkiv National University of Radio Electronics
ORCID: 0009-0006-1867-374X

METHOD FOR IMPROVING THE EFFICIENCY OF FUNCTIONING UNIX-LIKE SYSTEMS

The article proposes a method for improving the efficiency of the functioning of UNIX-shaped operating systems, aimed at ensuring stable and reliable operation of the server infrastructure of the corporate network in conditions of high load and limited hardware resources. This approach involves the use of automation mechanisms for the main system resources that contribute to the continuous operation of server infrastructure, as well as the implementation of optimization measures based on methods of monitoring and diagnostics using built -in system utilities. The proposed method allows you to detect and eliminate potential failures in a timely manner, prevent excessive use of resources and ensure stable and continuous operation of the system in high load conditions. The work provides recommendations on the automation of the processes of monitoring and management of resources, as well as the possibilities of integration of the proposed method with centralised control systems that use early detection tools in real time. The use of intellectual approaches to administration helps to increase the overall productivity, stability and energy efficiency of server infrastructure. Particular attention is paid to optimization measures that reduce the load on hardware resources, reduce the delays in large data processing and improve the energy efficiency of server infrastructure. The efficiency of implementation of the proposed method has been verified by modelling the overload of a server system of the local computer network of the Department of Electronic

Computing Machines of Kharkiv National University of Radio Electronics. The results have confirmed the expediency and efficiency of applying the proposed approach to ensure stable servers in modern corporate networks.

Key words: *method, automation, monitoring, reliability, server.*

Постановка проблеми

Розвиток інформаційно-комунікаційних технологій сучасного суспільства формує новий інформаційний простір, в якому цифрові технології значно переважають традиційні засоби обробки та передачі інформації. Проте це спричиняє експоненціальне зростання обсягів інформаційних ресурсів, функціонування яких забезпечується за рахунок розвиненої серверної інфраструктури [1]. В сучасних умовах мережна інфраструктура будується з використанням технологій віртуалізації та контейнеризації, що дозволяє створювати гнучкі, масштабовані та продуктивні серверні середовища, які виступають основою ресурсного забезпечення інформаційних систем [2].

Особливої актуальності набуває проблема забезпечення високої продуктивності серверних систем в умовах постійного зростання обсягів обчислювальних ресурсів для обробки даних. Забезпечення ефективної роботи таких систем також залежить від здатності операційної системи (ОС) сервера ефективно керувати доступними апаратними ресурсами [3].

Сучасні підходи до забезпечення продуктивності мережної інфраструктури, зокрема серверів, базуються на комплексному підході, що забезпечує інтеграцію методів моніторингу, аналізу та оптимізації системних ресурсів [4]. Використання вбудованих утиліт операційної системи для збору телеметричних даних в UNIX-подібних операційних системах дозволяє виявляти критичні події та «вузькі місця» у функціонуванні системи [5]. Отримані дані стають підґрунтям для забезпечення стабільної та безперервної роботи ОС, прогнозування та застосування оптимізаційних рішень, що дозволяють зменшити затримки в обробці запитів, знизити споживання ресурсів та виявляти критичні події на ранніх етапах, а також мінімізувати ручне керування шляхом впровадження автоматизації процесів.

Таким чином, забезпечення продуктивності розвиненої мережної інфраструктури, зокрема серверного середовища, вимагає потужного апаратного забезпечення та комплексного підходу щодо управління ресурсами, особливо в умовах обмеження апаратних ресурсів та динамічної зміни навантаження в сучасних корпоративних мережах.

Аналіз останніх досліджень і публікацій

Точкові дослідження в сфері продуктивності комп'ютерних мереж зосереджені на вирішенні ключових проблем, пов'язаних із забезпеченням високої продуктивності, стабільності, безпеки та масштабованості в умовах інтенсивного використання обчислювальних ресурсів. Активне впровадження технологій віртуалізації та контейнеризації, а також потреба в ефективному розгортанні ізольованих середовищ, зумовлюють необхідність постійного моніторингу та динамічного балансування навантажень для оптимального розподілу системних ресурсів [6, 7].

Зокрема, в роботі [8] наголошується на постійному зростанні енергоспоживання внаслідок перевантаження центрів обробки даних, зокрема в хмарних середовищах, – від 200 ТВт · год у 2016 році до прогнозованих 2967 ТВт · год у 2030 році. Автори підкреслюють важливість впровадження засобів віртуалізації та автоматизації управління ресурсами на різних рівнях, від програмного забезпечення до операційної системи та інфраструктури віртуалізації, з метою підвищення енергоефективності та зниження викидів CO_2.

Додатки, орієнтовані на обробку великих даних, потребують значних обчислювальних ресурсів, а також відповідного середовища для зберігання, обробки та аналітики інформації в розподілених системах. Це зумовлює експоненційне зростання навантаження на серверні потужності. Як зазначено в роботі [9], ефективне застосування контейнеризації, механізмів планування контейнерів, інструментів керування, автоматизованого розгортання та масштабування дозволяє суттєво знизити навантаження, водночас забезпечуючи балансування ресурсів, що є критично важливим при побудові масштабованих серверних архітектур.

Водночас, хоча вищезгадані фактори суттєво впливають на продуктивність серверного середовища, вони можуть бути менш критичними порівняно з проблемами, пов'язаними з внутрішньою неефективністю роботи самої операційної системи. До таких проблем належать накопичення тимчасових або зайвих файлів, неефективне управління фоновими процесами, надмірне споживання ресурсів окремими службами, а також відсутність оптимального розподілу ресурсів між системними компонентами.

В роботі [10] обґрунтовано, що одним з ефективних підходів до підвищення продуктивності операційних систем в серверному середовищі є впровадження ієрархічних моделей управління ресурсами, орієнтованих на багаторівневу оптимізацію. Зокрема, застосування планувальника рутинного технічного обслуговування (ROS) та модуля раннього виявлення збоїв (ETM) забезпечує проактивне управління системним станом. ROS виконує автоматизовану організацію циклічних завдань з очищення, діагностики та профілактики, знижуючи ризики накопичення помилок, тоді як ETM виконує безперервний моніторинг і виявлення аномалій на ранніх етапах. Комплексне використання цих компонентів сприяє підвищенню стабільності, енергоефективності та загальної продуктивності операційної системи, що є критично важливим для високонавантажених серверних інфраструктур.

Додатково, як показано в роботі [11], перспективним напрямом оптимізації є інтеграція інтелектуальних механізмів самоналаштування та автономного управління, що базуються на методах штучного інтелекту та машинного

навчання. Зважаючи, що існуючі програмні рішення здатні реагувати на збої та виконувати базове налаштування, їх автономність залишається обмеженою через відсутність гнучких механізмів моніторингу в реальному часі та недостатню реалізацію зворотного зв'язку. Це значно ускладнює реалізацію повністю адаптивних систем керування.

Впровадження системного моніторингу стану операційної системи, що підтверджено в роботах [12, 13], підкреслює ефективність використання вбудованих системних утиліт в операційних системах UNIX-подібного типу для контролю за станом ОС та споживанням ресурсів. Такий підхід дозволяє забезпечити високий рівень прозорості функціонування системи, мінімізуючи потребу в використанні стороннього програмного забезпечення, що є актуальним в середовищах з обмеженими ресурсами. Таким чином, використання стандартних інструментів моніторингу виступає важливим елементом в побудові ефективної моделі адміністрування серверних операційних систем.

Таким чином, аналіз останніх досліджень та публікацій [6–13] показує важливість розробки та впровадження інтелектуальних підходів до оптимізації функціонування операційних систем серверного рівня, які здатні забезпечити адаптивність до змінних умов експлуатації, проактивне виявлення та попередження збоїв, а також автоматизоване управління ресурсами з мінімальним втручанням з боку адміністратора. Інтеграція інструментів раннього виявлення аномалій, засобів інтелектуального моніторингу в реальному часі та використання вбудованих системних утиліт сприяє підвищенню загальної продуктивності, стабільності та енергоефективності серверних інфраструктур, що функціонують в умовах високого навантаження та обмежених ресурсів.

Формулювання мети дослідження

Метою дослідження є розробка методу підвищення ефективності функціонування операційних систем UNIX–подібного типу, орієнтованого на забезпечення стабільної, надійної та енергоефективної роботи серверної інфраструктури шляхом впровадження системного моніторингу, автоматизованого управління ресурсами та раннього виявлення збоїв.

Викладення основного матеріалу дослідження

Алгоритмичне забезпечення

Розроблений метод передбачає комплексний підхід до забезпечення стабільності, надійності та енергоефективності функціонування ОС, яке забезпечено поєднанням моніторингу, автоматизованого управління ресурсами та механізмів виявлення збоїв. Алгоритм складається з наступних етапів:

1. Ініціалізація системного моніторингу.

На цьому етапі здійснюється запуск скриптів, призначених для початкового збору та періодичного оновлення системної інформації щодо основних обчислювальних ресурсів, зокрема процесору, оперативної пам'яті та дискового простору. Збір метрик виконується через використання стандартних вбудованих утиліт UNIX–подібних систем, а зібрані дані зберігаються у системних лог-файлах, які створюються в окремому каталозі.

2. Автоматизоване управління ресурсами.

На другому етапі здійснюється автоматизоване управління ресурсами шляхом впровадження механізмів, що забезпечують технічне обслуговування та автоматичне реагування на події, які пов'язані з деградацією основних ресурсів UNIX–подібної операційної системи. Основним інструментом автоматичного управління ресурсами слугує вбудований планувальник завдань cron, який дозволяє виконувати скрипти з визначеним інтервалом без втручання адміністратора в фоновому режимі. За необхідністю в скрипти додаються логічні модулі не лише для регулярної перевірки, а в разі виявлення збою, модуль виконання автоматичного перезапуску встановлених сервісів, що є критично важливим для служб резервного копіювання або віддаленого доступу. Кількість логічних модулів в скрипті визначається рівнем навантаження на ресурси, специфікою сервісів, які обслуговує серверна інфраструктура, та цільовим призначенням моніторингу. Частота виконання певного модулю скрипта відповідає критичності задачі моніторинга, що дозволяє мінімізувати участь адміністратора в рутинному обслуговуванні системи, забезпечуючи зниження ризику помилок та підвищуючи стабільність роботи серверної інфраструктури в умовах безперервного навантаження.

3. Контроль системного журналювання.

На цьому етапі реалізується контроль системного журналювання через безперервний аналіз системних журналів і повідомлень ядра з метою виявлення аномалій, помилок та несанкціонованих дій в режимі реального часу або з визначеною мінімальною затримкою, що важливо для раннього виявлення або прогнозування збоїв програмного забезпечення та потенційних загроз безпеки. Виявлені інциденти фіксуються у звітному журналі та формують тригер для сповіщень адміністратору. Таким чином контроль журналювання сукупно з попередніми етапами дозволяє сформувати базовий рівень захисту інфраструктури та первинну діагностику виявлених збоїв.

4. Механізм сповіщення та реагування.

Впровадження механізмів сповіщення на цьому етапі дозволяє своєчасне реагування на виявлені збої або критичні помилки в роботі операційної системи, що мінімізує час простою та відновлення системи, знижує ймовірність втрати даних та забезпечує безперервність надання сервісів. На цьому етапі може бути впроваджена інтеграція з централізованою системою моніторингу, що застосовується в корпоративній мережі. Таким чином, організація оперативного інформування адміністратора створює передумови для впровадження автоматизованої

реакції на системні збої або потенційні загрози, що загалом підвищує надійність і безпеку серверної інфраструктури, а також стабільність функціонування всієї корпоративної мережі.

5. Циклічний аналіз та адаптація.

На цьому етапі виконується формування повноцінного циклу моніторингу, який складається з діагностики, реагування на подію, адаптацію системи до вжитих заходів та прогнозування поведінки системи до змін навантаження.

Псевдокод, наведений нижче, дозволяє більш точно описати логіку виконання методу. В ньому зазначені конкретні умови, цикли, виклики функцій та обробку даних, що важливо для розуміння послідовності операцій в запропонованому методі.

```
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
```

Лістинг 1. Псевдокод запропонованого методу

Оцінка ефективності розробленого методу

Загальна ефективність методу підвищення функціонування Unix–подібних операційних систем може бути визначена шляхом зваженої оптимізації ключових факторів, зокрема максимізації стабільності, мінімізації кількості збоїв системи, енергоспоживання та скорочення затримки реагування на події. Ефективність методу F_{eff} визначається у вигляді багатофакторної оптимізаційної задачі наступного вигляду:

$$\max_t E(t) = w_1 \cdot S(t) - w_2 \cdot R(t) - w_3 \cdot P(t) - w_4 \cdot L(t), \quad (1)$$

де w_1, w_2, w_3, w_4 – вагові коефіцієнти, які визначають пріоритетність відповідного критерію задачі.

Компоненти оптимізаційної задачі (1) представлені нижче:

1. Оцінка стабільності системи $S(t)$ визначається часом безперервної роботи системи., що максимізується шляхом продовження аптайму. Оцінка стабільності системи може бути обчислена як:

$$S(t) = \frac{T_{uptime}}{T_{total}}, \quad (2)$$

де T_{uptime} – тривалість безперервної роботи системи без перезавантаження або критичних збоїв, с; T_{total} – загальний час моніторингу, с.

2. Оцінка надійності системи $R(t)$ визначається кількістю збоїв або помилок, що виникають протягом часу спостереження. Зменшення кількості помилок є ключовим фактором підвищення загальної працездатності системи. Оцінка надійності системи може бути обчислена як:

$$R(t) = 1 - \frac{F(t)}{F_{\max}}, \quad (3)$$

де $F(t)$ – кількість зафіксованих збоїв або критичних помилок за час t, од.; $F_{\max}$ – максимально допустима кількість помилок у межах інтервалу спостереження, с.

3. Енергоефективність $P(t)$ характеризується рівнем енергоспоживання ОС під час роботи. Мінімізація енергоспоживання дозволяє знизити теплове навантаження та подовжити термін служби серверного обладнання. Енергоефективність може бути обчислена як:

$$P(t) = 1 - \frac{W(t)}{W_{\max}}, \quad (4)$$

де $W(t)$ – фактичне енергоспоживання системи на момент часу, Вт; $W_{\max}$ – максимально допустиме енергоспоживання системи, Вт.

4. Швидкість реагування $L(t)$ визначає затримку системи у відповідь на певні події, зокрема запуск скриптів, спрацювання тригерів моніторингу, помилки служб. Мінімізація затримки вказує на вищу оперативність системи. Швидкість реагування на події може бути обчислена як:

$$L(t) = 1 - \frac{D(t)}{D_{\max}}, \quad (5)$$

де $D(t)$ – фактична середня затримка системи у відповіді на події, мс; $D_{\max}$ – максимально допустима затримка реагування, мс.

Запропонований підхід дозволяє комплексно оцінити ефективність методу підвищення продуктивності Unix-подібних систем на основі ключових показників: стабільності, надійності, енергоефективності та швидкості реагування на події. Така модель забезпечує більш точну кількісну оцінку взаємозв'язку між ефективністю функціонування операційної системи та основними параметрами її роботи, надається відповідна вага.

Постановка експерименту

На початковому етапі було проведено аналіз апаратних вимог для операційної системи Ubuntu 18.04 LTS, яку було обрано в якості тестового середовища на лабораторному обладнанні кафедри електронних обчислювальних машин ХНУРЕ. Було визначено, що для забезпечення стабільної роботи обраної системи достатньо процесора, який складається з 2 ядер тактовою частотою 2,9 ГГц та 4 ГБ оперативної пам'яті. Дисковий простір складає 20 ГБ, який було надано операційній системі, забезпечує достатній обсяг для забезпечення стабільної роботи операційної системи, а також може бути використаний для розгортання необхідних сервісів, зокрема розгортання веб-сайту на базі вебсерверу Apache 2. Для передачі даних пропускна здатність каналу зв'язку становить не менше 1 Гбіт/с. В тестовому середовищі було створено як реальне навантаження у вигляді розгорнутого сайта, так і створено імітацію додаткового навантаження у вигляді фіктивних логів кожен розміром 200 Мб.

В рамках розробки методу підвищення ефективності функціонування ОС було створено та запущено автоматизований скрипт моніторингу стану операційної системи тестового середовища, який виконується за визначеним

розкладом за допомогою утиліти cron. На рис. 1 представлено узагальнену схему основних компонентів операційної системи, які підлягають моніторингу в рамках розробленого методу.

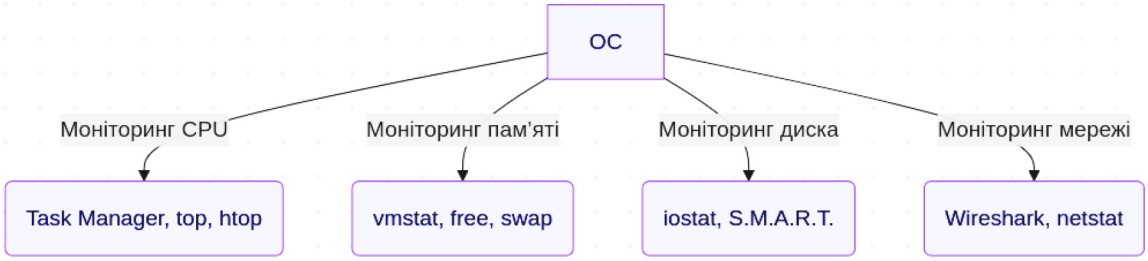

Рис. 1. Основні компоненти операційної системи, які підлягають моніторингу

Розроблений скрипт має модульну архітектуру, що дозволяє його адаптувати до вимог моніторингу, а також масштабувати, додаючи за потребою відповідні логічні блоки. Автоматизований скрипт охоплює основні компоненти операційної системи: CPU, оперативну пам'ять та дисковий простір. За результатами роботи скрипту додатково формуються сповіщення у вигляді повідомлень в месенджер адміністратора про критичні події або результати роботи скрипта, наприклад досягнення критичного рівня заповнення диску або видалення застарілих лог-файлів.

Архітектура скрипта містить наступні блоки:

1. Блок ініціалізації.

В цьому модулі визначено змінні середовища, перевірка прав доступу, створення лог-файлів та збереження результатів моніторингу, а також визначається поточний час виконання для формування звітів.

2. Блок збору метрик.

В цьому модулі виконується діагностика тестового середовища за допомогою вбудованих утиліт, при чому кожна команда запускається в окремому процесі, що забезпечує асинхронну обробку даних, забезпечуючи зниження затримки. Зокрема, утиліта free дозволяє отримати оперативні дані про використання оперативної пам'яті, що необхідно для виявлення потенційного дефіциту оперативної пам'яті та може призвести до зниження продуктивності. Утиліта vmstat забезпечує розширений моніторинг основних показників роботи ядра операційної системи, включаючи статистику процесів, навантаження на CPU, обмін пам'яттю, а також активність вводу/виводу. Це дозволяє оперативно виявити ознаки перевантаження або «вузькі місця» в роботі підсистеми. Утиліта htop використовується для загальної оцінки тривалості роботи системи та поточного стану активних процесів.

3. Блок обробки подій.

В цьому модулі виконується обробка критичних подій на основі виявлених підозрілих записів у лог-файлах, відбувається контроль заповнення дискового простору та очищення логів. Цей модуль видалення тимчасових та застарілих файлів, що зменшує навантаження на файлову систему та прискорює доступ до активних даних. Контроль обсягу дискового простору виконується з використанням утиліти df для оцінка зайнятого простору на розділах та утіліти du для аналізу обсягу каталогів, що дає змогу створити порогові правила для виявлення перевантажених розділів. В якості порогового значення дискового простору було обрано заповнення диску на 80 %, після чого скрипт виконує очищення кешу та тимчасових логів. Перевірка та перезапуск деградованих сервісів, зокрема Apache2, MySQL, SSH, для роботи вебсайту виконується через використання утиліти systemctl. В разі виявлення збою виконується автоматичний перезапуск сервісів, що додано до скрипту.

4. Блок сповіщень.

В разі виявлення інцидентів, зокрема новий стан системи, заповнення або очищення диску активується механізм сповіщень адміністратору (рис. 2).

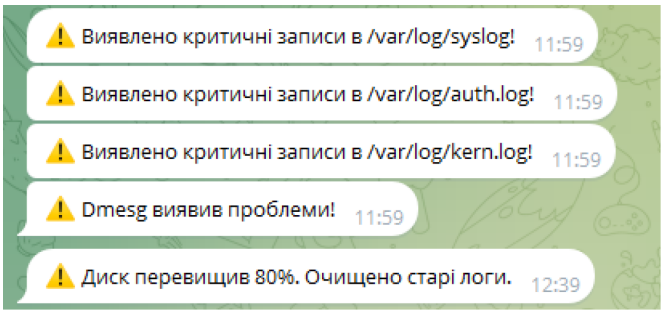

Рис. 2. Сповіщення в месенджері адміністратора про результати роботи автоматизованого скрипту

5. Блок логування та збереження статистики.

Зібрані метрики записуються в локальні файли з відповідним найменуванням. За результатами роботи скрипта за обраним часовим розкладом формується зведений звіт про навантаження за визначений період часу.

6. Блок автоматичного виконання.

В цьому модулі роботу скрипта інтегровано з планувальником подій – утилітою cron для запуску моніторингу операційної системи. Для тестування було обрано наступні періоди проведення моніторингу:

Monitor – виконання збору метрик та запис до лог-файлів кожні 5 хвилин про стан системних процесів;

daily – щоденне очищення тимчасових файлів, очищення диску та контроль системних служб;

hourly – погодинна перевірка логів на критичні записи та надсилання сповіщень про стан системи;

weekly – щотижневий звіт стану CPU.

Для інтеграції з планувальником подій цей скрипт було додано до конфігураційного файлу у вигляді відповідного запису, що формує політики моніторингу та формування звітів.

За результатами проведених тестувань роботи автоматизованого скрипту для моніторингу стану операційної системи було отримано наступні результати зміни досліджуваних показників операційної системи Ubuntu 18.04 (рис. 3).

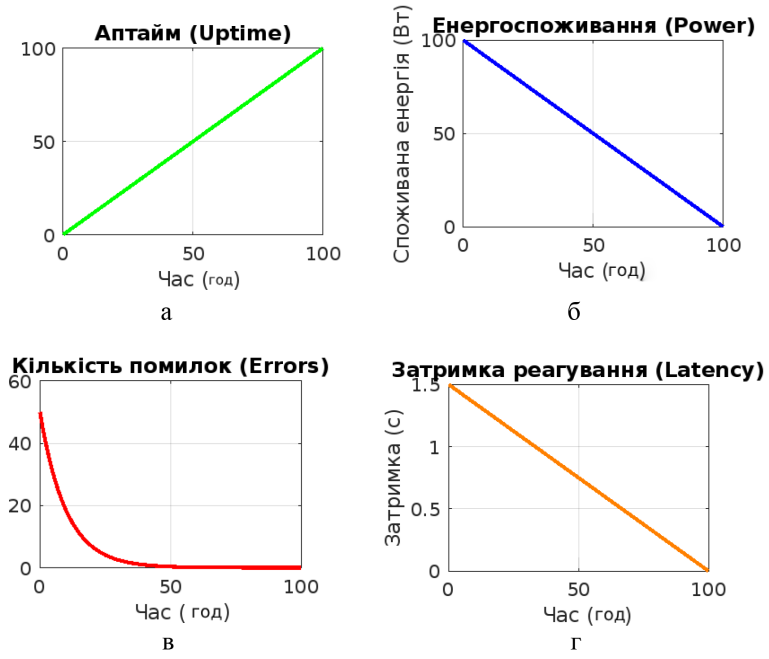

Рис. 3. Зміни показників ОС Ubuntu 18.04 LTS за часом: *а*) час аптайму; *б*) енергоспоживання ОС; *в*) кількість виявлених та виправлених помилок роботи; г) затримка реагування на події операційною системою

Висновки

В даній роботі було запропоновано та детально проаналізовано метод підвищення ефективності функціонування Unix-подібних операційних систем серверної інфраструктури. Запропонований метод базується на використанні автоматизованого скрипту для моніторингу системних показників в реальному часі, що дає змогу забезпечити стабільну та безперервну роботу ОС з мінімальним втручанням адміністратора. Метод дозволяє автоматизувати контроль стану системи, оркестрацію основних системних процесів, зменшуючи вплив людського фактора та підвищуючи надійність роботи. Наукова новизна полягає у впровадженні адаптивного механізму моніторингу, який фіксує поточний стан системи та забезпечує можливість прогнозування подальшого функціонування під час зростання навантаження, що дозволяє динамічне балансування між продуктивністю, енергоспоживанням та безпекою.

Проведене моделювання показало, що автоматизація та постійний моніторинг призводять до зниження латентності, кількості помилок та енергоспоживання, водночас підвищуючи аптайм системи, що сприяє підвищенню ефективності роботи серверу, та забезпечує довготривалу роботу та стабільність ОС, знижуючи ризики збоїв та мінімізуючи втручання адміністратора.

Таким чином, запропонований метод є доцільним для використання в серверній інфраструктурі корпоративних комп’ютерних мереж в умовах постійної зміни навантаження або в обмеженні апаратних ресурсів. Подальші дослідження доцільно зосередити на інтеграції запропонованого методу з системами інтелектуального моніторингу, який застосовується в сучасних корпоративних комп’ютерних з мережах для створення автономних систем адміністрування.

Список використаної літератури

1. Peñalvo, F. J. G., Sharma, A., Chhabra, A., Singh, S. K., Kumar, S., Arya, V., & Gaurav, A. (2022). Mobile cloud computing and sustainable development: Opportunities, challenges, and future directions. International Journal of Cloud Applications and Computing (IJCAC), 12(1), 1–20.
2. Arogundade, O. R., & Palla, K. (2023). Virtualization revolution: Transforming cloud computing with scalability and agility.
3. Apriani, D., Afrijaldi, R., Auliya, N., & Darmawan, A. A. (2024). Operating System and Server Integration for Business Effectiveness. IAIC Transactions on Sustainable Digital Innovation (ITSDI), 5(2), 91–99. https://doi.org/10.34306/itsdi.v5i2.615
4. Шостак, А. Р., Ткачов, В. М. (2023). Питання оптимізації бізнес-процесів з використанням сучасних інформаційних технологій.
5. Duin, J., McKeown, S., & Abubakar, M. (2024, June). Exploring DTrace as an Incident Response Tool for Unix Systems. In The International Conference on Cybersecurity, Situational Awareness and Social Media (pp. 169–193). Singapore: Springer Nature Singapore.
6. Queiroz, R., Cruz, T., Mendes, J., Sousa, P., & Simões, P. (2023). Container-based virtualization for real-time industrial systems – a systematic review. ACM Computing Surveys, 56(3), 1–38.
7. Martin, E. (2022). Virtualization and containerization: a new concept for data center management to optimize resources distribution.
8. Katal, A., Dahiya, S. & Choudhury, T. *Energy efficiency in cloud computing data centers: a survey on software technologies*. Cluster Comput 26, 1845–1875 (2023). https://doi.org/10.1007/s10586–022–03713–0
9. Singh, N., Hamid, Y., Juneja, S. et al. *Load balancing and service discovery using Docker Swarm for microservice based big data applications*. J Cloud Comp 12, 4 (2023). https://doi.org/10.1186/s13677–022–00358–7
10. Papaioannou, C.; Dimara, A.; Papaioannou, A.; Tzitzios, I.; Anagnostopoulos, C.–N.; Krinidis, S. Hierarchical Resources Management System for Internet of Things–Enabled Smart Cities. Sensors 2025, 25, 616. https://doi.org/10.3390/s25030616
11. Gill, S. S., Xu, M., Ottaviani, C., Patros, P., Bahsoon, R., Shaghaghi, A.,. Uhlig, S. (2022). AI for next generation computing: Emerging trends and future directions. Internet of Things, 19, 100514.
12. Umer, M. (2023). Architecture and Design of the Linux Storage Stack. Packt Publishing Ltd.
13. Davis, J. (2022). Modern System Administration. O'Reilly Media, Inc.

References

1. Peñalvo, F. J. G., Sharma, A., Chhabra, A., Singh, S. K., Kumar, S., Arya, V., & Gaurav, A. (2022). Mobile cloud computing and sustainable development: Opportunities, challenges, and future directions. International Journal of Cloud Applications and Computing (IJCAC), 12(1), 1–20.
2. Arogundade, O. R., & Palla, K. (2023). Virtualization revolution: Transforming cloud computing with scalability and agility.
3. Apriani, D., Afrijaldi, R., Auliya, N., & Darmawan, A. A. (2024). Operating System and Server Integration for Business Effectiveness. IAIC Transactions on Sustainable Digital Innovation (ITSDI), 5(2), 91–99. https://doi.org/10.34306/itsdi.v5i2.615
4. Shostak, A. R., Tkachov, V. M. (2023). Pytannia optymizatsii biznes-protsesiv z vykorystanniam suchasnykh informatsiinykh tekhnolohii.
5. Duin, J., McKeown, S., & Abubakar, M. (2024, June). Exploring DTrace as an Incident Response Tool for Unix Systems. In The International Conference on Cybersecurity, Situational Awareness and Social Media (pp. 169–193). Singapore: Springer Nature Singapore.
6. Queiroz, R., Cruz, T., Mendes, J., Sousa, P., & Simões, P. (2023). Container-based virtualization for real-time industrial systems – a systematic review. ACM Computing Surveys, 56(3), 1–38.
7. Martin, E. (2022). Virtualization and containerization: a new concept for data center management to optimize resources distribution.
8. Katal, A., Dahiya, S. & Choudhury, T. *Energy efficiency in cloud computing data centers: a survey on software technologies*. Cluster Comput 26, 1845–1875 (2023). https://doi.org/10.1007/s10586–022–03713–0
9. Singh, N., Hamid, Y., Juneja, S. et al. *Load balancing and service discovery using Docker Swarm for microservice based big data applications*. J Cloud Comp 12, 4 (2023). https://doi.org/10.1186/s13677–022–00358–7
10. Papaioannou, C.; Dimara, A.; Papaioannou, A.; Tzitzios, I.; Anagnostopoulos, C.–N.; Krinidis, S. Hierarchical Resources Management System for Internet of Things–Enabled Smart Cities. Sensors 2025, 25, 616. https://doi.org/10.3390/s25030616
11. Gill, S. S., Xu, M., Ottaviani, C., Patros, P., Bahsoon, R., Shaghaghi, A.,. Uhlig, S. (2022). AI for next generation computing: Emerging trends and future directions. Internet of Things, 19, 100514.
12. Umer, M. (2023). Architecture and Design of the Linux Storage Stack. Packt Publishing Ltd.
13. Davis, J. (2022). Modern System Administration. O'Reilly Media, Inc.