УДК 004.451.25

В. А. Шмалюх
Е. С. Вознюк
Б.С. Білецький

КРОСПЛАТФОРМНИЙ МОНІТОРИНГ РЕСУРСІВ ОС НА PYTHON З ВИКОРИСТАННЯМ PSUTIL

Вінницький національний технічний університет

Анотація

У роботі розглянуто методи та інструменти моніторингу системних ресурсів операційної системи засобами мови програмування Python. Проаналізовано можливості бібліотек psutil, py-cpuinfo, GPUtil для отримання інформації про завантаження центрального процесора, використання оперативної пам'яті, дискових операцій, мережевої активності та управління процесами. Досліджено архітектуру побудови систем моніторингу в реальному часі з візуалізацією даних за допомогою бібліотеки matplotlib. Розроблено практичну реалізацію модульної системи моніторингу з можливостями сповіщення при перевищенні порогових значень, логування даних та формування звітів. Результати демонструють ефективність використання Python для створення кросплатформових рішень системного аналізу.

Ключові слова: системний моніторинг, Python, psutil, управління процесами, CPU, RAM, системні ресурси, моніторинг у реальному часі, системне програмування.

Abstract

The paper examines methods and tools for monitoring system resources of the operating system using the Python programming language. The capabilities of the psutil, py-cpuinfo, GPUtil libraries for obtaining information about the CPU load, RAM usage, disk operations, network activity, and process management are analyzed. The architecture for building real-time monitoring systems with data visualization using the matplotlib library is studied. A practical implementation of a modular monitoring system with notification capabilities when threshold values are exceeded, data logging, and report generation is developed. The results demonstrate the effectiveness of using Python for creating cross-platform system analysis solutions.

Keywords: system monitoring, Python, psutil, process management, CPU, RAM, system resources, real-time monitoring, system programming.

Вступ

Моніторинг системних ресурсів є важливим аспектом адміністрування серверів, оптимізації продуктивності застосунків та забезпечення стабільності IT-інфраструктури. Сучасні операційні системи надають численні засоби для отримання інформації про стан апаратних і програмних компонентів, однак їх використання часто вимагає знання специфічних API різних платформ (Windows API, Linux /proc файлова система, системні виклики в macOS) [1-3]. Це ускладнює створення кросплатформних програм для моніторингу системних ресурсів.

Python є ефективною мовою для системного програмування в цьому аспекті, завдяки наявності бібліотек, що надають уніфікований інтерфейс для роботи з системними ресурсами. Бібліотека «psutil» (process and system utilities) є стандартом для кросплатформного моніторингу системи, підтримуючи Linux, Windows, macOS, FreeBSD, OpenBSD, NetBSD, Sun Solaris та AIX [4, 5].

Актуальність дослідження зумовлена зростаючими вимогами до автоматизації DevOps-процесів, необхідністю виявлення проблем продуктивності та оптимізації використання хмарних ресурсів (де оплата часто прив'язана до споживання CPU/RAM) та побудови систем діагностики для високонавантажених застосунків [6, 7].

Мета роботи – дослідити можливості Python для створення системи моніторингу ресурсів ОС, розробити модульну архітектуру системи моніторингу, продемонструвати практичні підходи до збору, обробки та аналізу метрик CPU, пам'яті, дисків та процесів.

Результати дослідження

Для оцінки можливості використання Python як інструмента системного моніторингу було виконано збір та аналіз основних метрик роботи комп’ютера в реальному часі. Для цього було розроблено програму для моніторингу системних ресурсів комп’ютера. Програма має модульну структуру та

складається з кількох логічних частин, кожна з яких відповідає за окремий етап роботи: збір даних, їх аналіз, збереження результатів, сповіщення про перевищення допустимих значень та візуалізацію інформації.

Для збору інформації про стан системи використовується бібліотека psutil, яка дозволяє отримувати дані про роботу процесора, оперативної пам'яті, дисків, мережі та активних процесів [1]. Збір метрик виконується з фіксованим інтервалом часу, що дає можливість спостерігати зміну навантаження системи в динаміці. Обрані метрики дозволяють оцінити загальний стан системи та виявити потенційні проблеми продуктивності.

Програма визначає такі основні показники:

1. завантаження процесора в цілому та по окремих ядрах;
2. використання оперативної пам'яті та swap-пам'яті;
3. стан дискових розділів і операції читання та запису;
4. мережеву активність;
5. список активних процесів із найбільшим навантаженням на CPU та пам'ять.

Також у програмі реалізовано перевірку порогових значень для основних ресурсів. У випадку перевищення допустимого рівня навантаження система формує попереджувальні повідомлення, які відображаються у консолі.

У наведеному прикладі (рис. 1) зафіксовано повідомлення про високе використання дискового простору, коли заповнення диска перевищило допустимий рівень у 90%.

Після кожного циклу збору даних у консоль виводиться коротка зведена інформація, яка включає: поточне завантаження процесора у відсотках, відсоток використання оперативної пам'яті та кількість активних процесів у системі. Після завершення моніторингу програма повідомляє про збереження результатів у файлі та автоматично формує текстовий звіт (рис. 2).

У звіті відображається період моніторингу, кількість зібраних зразків, а також середні, мінімальні та максимальні значення навантаження процесора і пам'яті за час роботи програми. Додатково виводиться базова інформація про систему, зокрема кількість ядер процесора, обсяг оперативної пам'яті та кількість активних процесів.

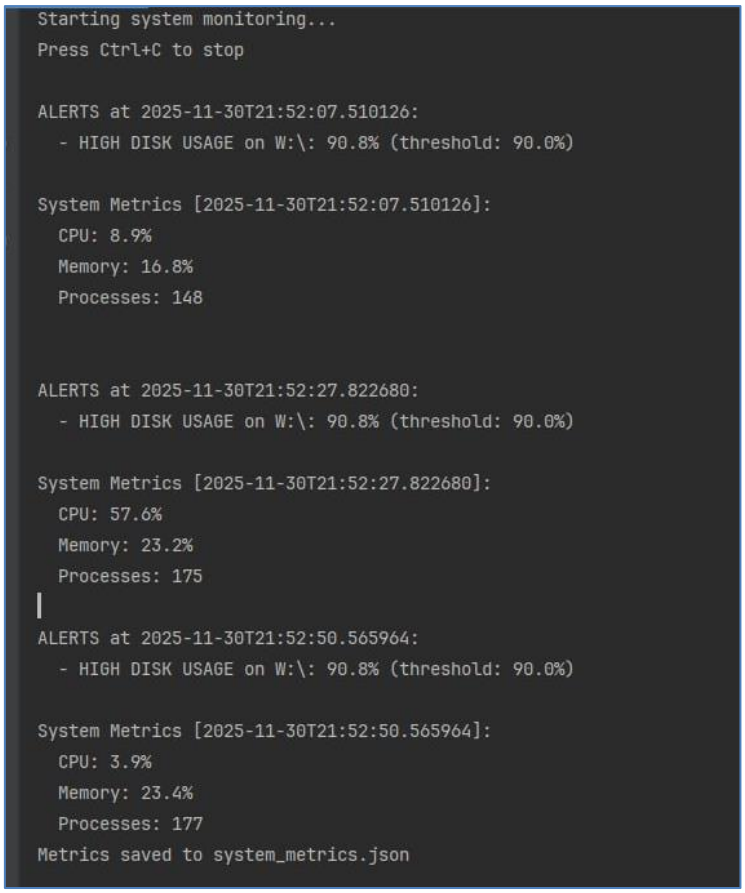

Рис. 1. Результат моніторингу системних ресурсів на Python (основна інформація + алерти)

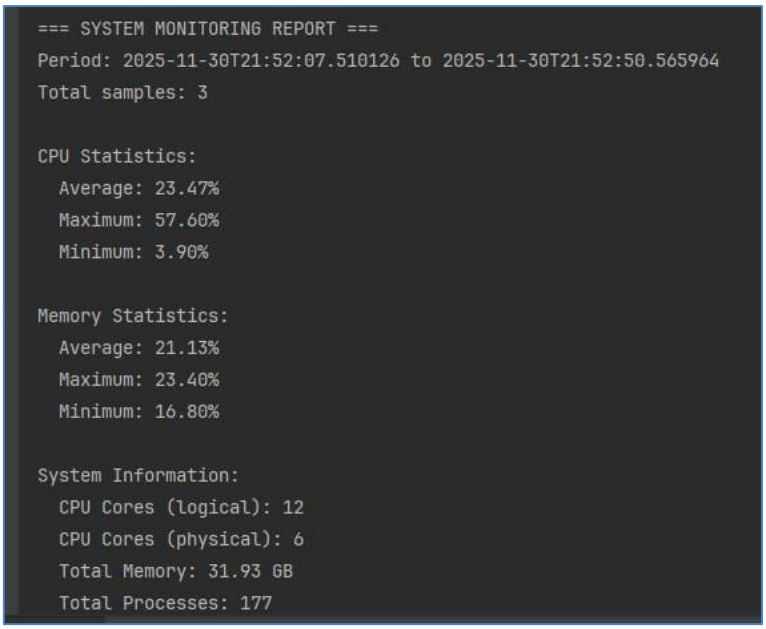

Рис. 2. Результат моніторингу системних ресурсів на Python (зведена інформація)

Всі зібрані дані зберігаються у файлі system_metrics.json у форматі JSON (рис. 3). Такий формат збереження даних дозволяє виконувати подальший аналіз результатів. Файл містить масив записів, де кожен запис відповідає одному циклу збору інформації про стан системи.

Кожен елемент у JSON-файлі містить такі основні блоки:

1. Час збору даних, для кожного запису зберігається мітка часу timestamp, яка показує, коли саме були зняті метрики.
2. Дані про процесор (CPU), у цьому блоці зберігається: загальне завантаження процесора у відсотках; завантаження кожного ядра окремо; кількість логічних та фізичних ядер; поточна, мінімальна та максимальна частота процесора; статистика роботи CPU (кількість перемикань контексту та переривань); середнє навантаження системи за 1, 5 та 15 хвилин (для Unix-подібних систем).
3. Дані про оперативну пам’ять, JSON-файл містить інформацію про: загальний обсяг оперативної пам’яті; використану та доступну пам’ять; відсоток використання RAM; обсяг та використання swap-пам’яті.
4. Дані про дискову підсистему, для кожного дискового розділу зберігається: назва пристрою та точка монтування; тип файлової системи; загальний обсяг, використаний та вільний простір; відсоток заповнення диска. Також зберігається загальна статистика операцій читання та запису (кількість операцій, байтів і час доступу).
5. Дані про мережу, у цьому блоці фіксується: кількість переданих та отриманих байтів; кількість пакетів; кількість помилок передачі; кількість активних мережевих з’єднань.
6. Дані про процеси, JSON-файл містить: загальну кількість активних процесів у системі; список процесів з найбільшим навантаженням на CPU; список процесів з найбільшим використанням оперативної пам’яті. Для кожного процесу зберігається ідентифікатор процесу, ім’я, користувач, стан, відсоток CPU та RAM.

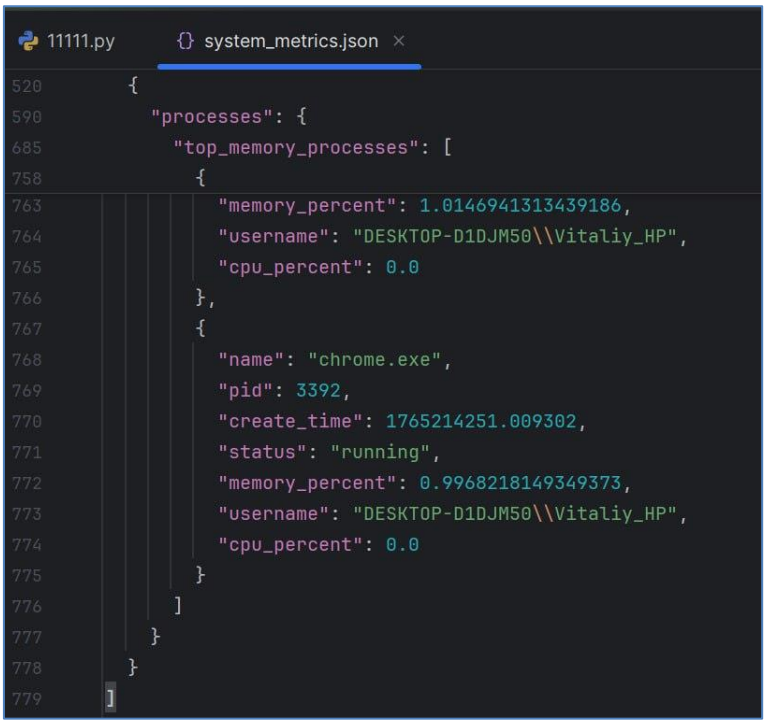

Рис. 3. Частина вмісту файлу JSON з повними результатами моніторингу

Під час виконання моніторингу було проаналізовано вплив процесу збору метрик на загальне завантаження системи. У ході спостереження було зафіксовано різкі, проте не постійні стрибки навантаження процесора. Короткочасні навантаження процесора найбільш ймовірно зумовлені фоновими процесами операційної системи та не мають стабільного характеру. Бібліотека psutil для збору системних метрик не створює суттєвого додаткового навантаження та може застосовуватися для регулярного контролю стану системи.

Реалізовано додаткову модифікацію програми, що дозволяє визначати найбільш навантажені ресурси (рис. 4). Це допомагає відстежувати які саме процеси навантажують центральний процесор (CPU), за потреби їх можна завершити, що забезпечить більшу продуктивність системи. "Оскільки користувач не завжди може визначити, чи є процес справді важливим і чи його зупинка не порушить роботу системи, у разі спроби завершити системний процес запит буде відхилено, а користувача буде повідомлено відповідним сповіщенням.

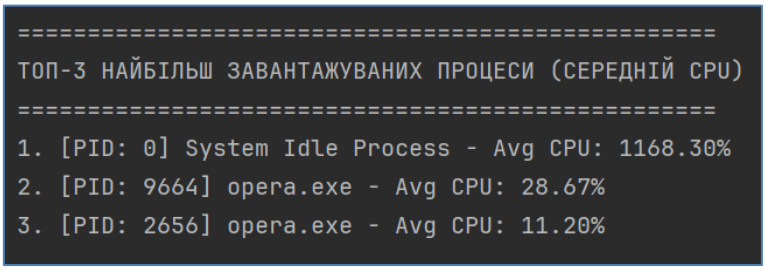

Рис. 4. Демонстрація найбільш завантажуваних процесів

Архітектура розробленої системи складається з чотирьох основних блоків (рис. 5). На рівні збору даних програма отримує інформацію про стан системи за допомогою бібліотеки psutil [4]. Далі ці дані передаються на рівень обробки, де виконується перевірка значень та формування попереджень. Рівень збереження відповідає за накопичення результатів у файлах формату JSON, що дозволяє зберігати історію вимірювань. На рівні представлення користувач може переглядати результати у вигляді текстового виводу або використовувати їх для подальшого аналізу та візуалізації. Запропонована структура підтверджує можливість побудови простих і розширюваних систем моніторингу засобами Python.

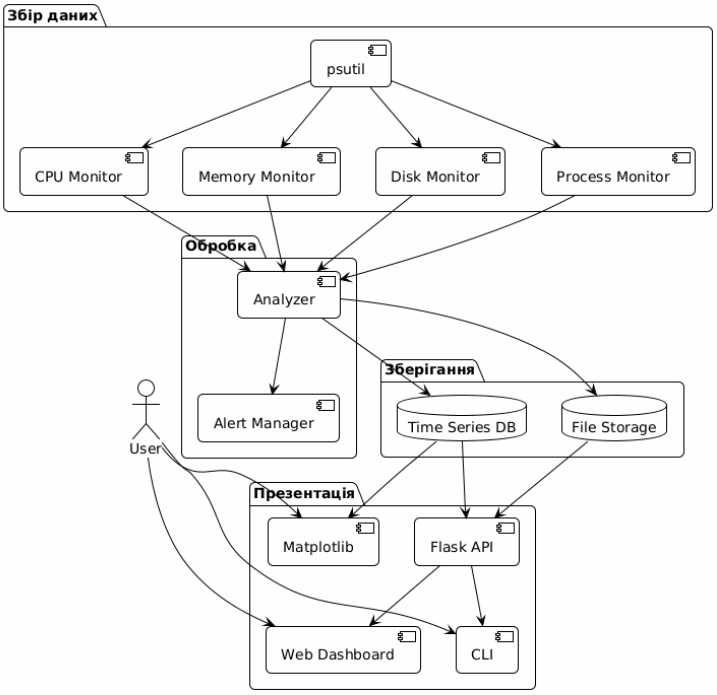

Рис. 5. Архітектура системи моніторингу системних ресурсів на Python

Висновки

У роботі досліджено можливості використання мови програмування Python для моніторингу системних ресурсів комп'ютера. Показано, що завдяки бібліотеці psutil можна отримувати ключові метрики стану системи (завантаження CPU, використання оперативної пам'яті, стан дискової підсистеми, мережеву активність та інформацію про процеси) незалежно від операційної системи.

Було розроблено програму для моніторингу системних ресурсів з модульною структурою, що включає етапи збору, аналізу та збереження даних, а також механізм формування попереджувальних повідомлень при перевищенні порогових значень. Практичні результати підтвердили, що процес збору метрик може використовуватися для регулярного контролю стану системи.

Отримані результати свідчать про доцільність застосування Python і бібліотеки psutil для створення кросплатформних систем моніторингу.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Microsoft. Windows API Index. [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list.
2. Linux Foundation. The /proc Filesystem. [Електронний ресурс] – Режим доступу: https://www.kernel.org/doc/html/latest/filesystems/proc.html.
3. Apple Inc. sysctl(3) — BSD System Calls Manual. [Електронний ресурс] – Режим доступу: https://developer.apple.com/library/archive/documentation/System/Conceptual/ManPages_iPhoneOS/man3/sysctl.3.html.
4. Giampaolo Rodolà. "psutil documentation". ReadTheDocs, 2023. [Електронний ресурс] – Режим доступу: https://psutil.readthedocs.io/.
5. Love, R. "Linux System Programming: Talking Directly to the Kernel and C Library". O'Reilly Media, 2nd Edition, 2013.
6. Google. "Site Reliability Engineering: How Google Runs Production Systems". O'Reilly Media, 2016.
7. Amazon Web Services. AWS Well-Architected Framework – Cost Optimization Pillar. [Електронний ресурс] – Режим доступу: https://docs.aws.amazon.com/wellarchitected/latest/cost-optimization-pillar/.

Шмалюх Віталій Анатолійович – студент групи 1ПКТ-24б, кафедра комп'ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: shmaliukhvitaliy@gmail.com

Вознюк Евеліна Сергіївна - студентка групи 1ПКТ-24б, кафедра комп'ютерних наук, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: evelinvozno@gmail.com

Білецький Богдан Сергійович – кандидат технічних наук, асистент кафедри комп'ютерних систем управління, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м. Вінниця, e-mail: bohdanbeletskyi@gmail.com

Shmaliukh Vitalii – student of 1PKT-24b group, Department of Computer Science, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: shmaliukhvitaliy@gmail.com

Vozniuk Evelina - student of 1PKT-24b group, Department of Computer Science, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: evelinvozno@gmail.com

Biletskyi Bohdan – PhD, Assistant at the Department of Computer Control Systems, Faculty of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: bohdanbeletskyi@gmail.com