DOI 10.34185/1562-9945-5-154-2024-15
УДК 004.6:004.4:004.942

Н.А. Гук, М.Ю. Мітіков

СУЧАСНІ ПРОБЛЕМИ ІДЕНТИФІКАЦІЇ АНОМАЛІЙ В РОБОТІ ENTERPRISE SYSTEMS

Анотація. У сучасних корпоративних системах зростаюча складність і масштаби програмного забезпечення створюють нові виклики для підтримки їх надійної та стабільної роботи. Одним з критичних аспектів є ідентифікація аномалій, які можуть призвести до збоїв у роботі систем, втрати даних або зниження продуктивності. Виявлення таких аномалій на ранніх етапах є важливою задачею для забезпечення безперебійного функціонування підприємства.

Метою цієї публікації є наочно описати наявні виклики, складність програмних систем, непередбачуваність виникнення збоїв, унікальність кожної окремої системи та неможливість запису усієї необхідної інформації для знаходження першопричини.

Одним з ефективних методів діагностики аномалій є аналіз знімків пам'яті (memory dumps). Цей підхід дозволяє отримати детальний стан системи на момент виникнення помилки або аномальної поведінки, що значно спрощує процес виявлення першопричин. Проте, сучасні системи працюють у складних умовах з величезними обсягами даних, тому класичні підходи до аналізу часто виявляються недостатньо ефективними.

У даній статті розглядаються сучасні проблеми ідентифікації аномалій у роботі корпоративних систем за допомогою аналізу знімків пам'яті, аналізуються методи та інструменти, що використовуються для цього завдання, та обговорюються можливі напрями їх подальшого розвитку.

Ключові слова: ідентифікація аномалій, корпоративні системи, знімки пам'яті, оптимізація продуктивності, виявлення проблем, машинне навчання, профілювання коду, алгоритми оптимізації, моніторинг систем, розподілені системи.

Вступ. З кожним роком корпоративні системи стають все більш складними та масштабними, що збільшує кількість потенційних проблем, які можуть вплинути на їхню продуктивність і стабільність [1, 2, 3]. Аномалії в роботі таких систем, як збої в роботі програмного забезпечення або неочікувані відхилення від нормальної поведінки, можуть призводити до значних втрат продуктивності або навіть до повної зупинки роботи компанії. Виявлення та усунення цих аномалій є ключовим аспектом для забезпечення безперервного функціонування корпоративних систем.

Метою статті є дослідження сучасних алгоритмів та інструментів для виявлення аномалій у роботі корпоративних систем за допомогою аналізу знімків пам'яті, а також оцінка їх ефективності, викликів та обмежень. Стаття також спрямована на вивчення існуючих ме-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

тодів оптимізації продуктивності програмного забезпечення і визначення можливих напрямків подальших досліджень для покращення процесів ідентифікації та усунення проблем у корпоративних системах.

Алгоритми та інструменти для виявлення проблем. В основі статистичних методів лежить ідея математичної репрезентації даних, як-то середніх значень, дисперсій та інших статистичних показників [4]. Це дозволяє виявляти аномалії на рівні агрегованих даних, зазвичай в метриках системного моніторингу.

Важливо зазначити, що метод може лише використовуватись постфактум для виявлення факту аномалії, але не може надати першопричину яка призвела до проблемної поведінки.

Алгоритми машинного навчання можуть бути використані для автоматичного виявлення аномалій в системі. Вони дозволяють моделювати нормальне поведінку системи і виявляти відхилення від неї. Для виявлення аномалій використовується історична база метрик моніторингу [5]. Дозволяє ідентифікувати факт настання неординарної ситуації, але не надає першопричини.

Для детальної діагностики використовують знімки пам'яті та інші види профілювання [6]. Це може допомогти в ідентифікації вузьких місць на рівні коду, включаючи витоки пам'яті або неоптимальне використання ресурсів.

Лімітацією цього підходу є необхідність збору даних саме у час настання неочікуваної ситуації.

Біхевіоральні алгоритми спрямовані на аналіз "поведінки" системи або її компонентів. Це може включати в себе аналіз послідовностей операцій, швидкість обробки запитів, інтеракцій між компонентами та інше.

Загалом, ефективність виявлення проблем залежить від комбінації алгоритмічних методів, інструментарію, та їх правильної інтеграції в архітектуру системи.

Методи оптимізації та їх застосування. Методи оптимізації в програмній інженерії охоплюють широкий спектр технік, що мають на меті покращення ефективності, швидкодії, та стабільності систем [7]. Застосування цих методів може суттєво відрізнятися в залежності від домену, мови програмування, та архітектурних рішень.

Процедурна оптимізація стосується низькорівневих алгоритмічних підходів, які включають оптимізацію циклів, уникнення дублювання виразів, та використання оптимізованих структур даних. Тут може бути застосовано, наприклад, бітові маніпуляції для прискорення операцій.

Методи векторизації та паралелізації зосереджені на оптимізації для конкретного обладнання, використовуючи багатопоточність або специфічні інструкції процесора. Це дозволяє значно підвищити швидкодію, але може обмежувати крос-платформеність [8].

Однією з широко застосовуваних технік є кешування результатів функцій та запитів до бази даних, що значно знижує навантаження та прискорює відгук системи. Меморізація дозволяє зберігати результати важких функцій для подальшого повторного використання.

Профілювання коду дає можливість ідентифікувати "вузькі місця" в системі, де оптимізація принесе найбільший ефект. Моніторинг дозволяє відстежувати стан системи в реальному часі, ідентифікувати та реагувати на можливі проблеми.

В області роботи з базами даних основні методи оптимізації включають в себе правильний вибір індексів, оптимізацію запитів SQL, та денормалізацію даних.

Виклики та обмеження. В основі виявлення проблем в системах програмного забезпечення лежить агрегована діагностична інформація, яка може бути отримана з двох основних джерел: системних журналів та доступних метрик. Специфічні виклики виникають у контексті навантаження на систему та валідності отриманих даних.

Співвідношення між діагностичною ефективністю та навантаженням на систему, а також між глибиною аналізу та обсягом зібраної інформації, формує комплексні виклики в розробці алгоритмів та інструментів для виявлення проблем в програмних системах. Активний запис діагностичних даних може суттєво вплинути на загальну продуктивність системи [9]. Це може відбуватися через IO-навантаження або навіть викликати вибіркову втрату інформації через наявні механізми семплювання. Загальнодоступні метрики, хоча і можуть служити індикаторами проблем (наприклад, зростання використання пам'яті або CPU), рідко забезпечують достатній контекст для ідентифікації первинних причин.

Зазвичай, аналітичні методи включають в себе візуалізацію агрегованих даних, формулювання гіпотез на їх основі [6], та інкрементальний збір додаткової діагностичної інформації для перевірки цих гіпотез. Ідеальним сценарієм є можливість репродукування проблемного сценарію в ізольованому тестовому середовищі на основі зібраних даних. Але для побудови відповідної моделі необхідно використовувати реальні дані.

Прикладом є використання хеш-алгоритму який видає рівний розподіл даних на площині потенційних значень і згідно математичної моделі та тестових даних є ідеальним кандидатом. Але на практиці, система працює лише з вузким діапазоном даних і «теоритично-обґрунтований» алгоритм хешування видає результати на порядок гірший за алгоритм який видає рівний розподіл тільки для використовуваного діапазону.

Однією з інноваційних технік є використання знімків пам'яті від реальних процесів для аналізу та виявлення аномалій. Ці знімки можуть надати глибокий інсайт про стан системи у моменти критичних збоїв або падіння продуктивності. Враховуючи велику кількість обчислювальних операцій яку виконує система, зняття знімку із затримкою у декілька секунд буде містити інформацію яка на декілька мільярдів розрахунків знаходиться далі у часі і може не охоплювати дані які призвели до аномалії.

Труднощі при аналізі та оптимізації програмного забезпечення. Сучасні високонавантажені програмні системи є вкрай складними, що ускладнює аналіз та вибір оптимальних рішень. Множина залежностей між компонентами може призвести до неочікуваних побічних ефектів під час оптимізації. Необхідність отримання доступності системи близької до ста відсотків та незалежного масштабування частин під навантаженням накладає необхідність

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

рознесення частин по фізично-різним обчислювальним машинам, що означає недостатність розгляду одного програмного процесу для розуміння загальної поведінки системи.

Наявність теоретичного оптимального алгоритму не є гарантом прийнятної сукупної швидкодії. Наприклад, система без вільних потоків для запуску частини розрахунків у алгоритмах з паралелізацією призведе до стагнації програмного додатку.

При аналізу цього інциденту не буде жодних діагностичних відміток щодо реальної причини бо його першопричина є механізм розподілу потоків для виконання – частин керуючого середовища яка знаходиться поза полем зору інженерів програмного коду.

Навіть відкинувши непередбачувану природу явища, що унеможливлює запуск короткої сесії профілювання, а також високе навантаження системи що не гарантує можливість запуску нового процесу, в отриманій сесії розподіл використаного процесорного часу буде цілком позначено на алгоритмі.

Огляд обмежень поточних методів та інструментів. Вищеописані приклади наочно демонструють обмеження поточних методів:

– непередбачувана у часі природа явища унеможливлює завчасний запуск профілювання,

– високонавантажені системи у пікові навантаження мають дефіцит процесорного часу, успіх запуску профілювання (нового процесу) не гарантовано,

– непередбачуваність за джерелом унеможливлює запис у діагностичний лог саме тих даних які дозволять зрозуміти та чітко змоделювати проблему у тестовому середовищі,

– вплив середовища виконання (як менеджмент потоків) теж залишається поза увагою моделювання,

– жодна з існуючих технологій не дає 100% часу онлайн, то доступність розподіленої системи деградує від кількості взаємозалежних вузлів,

– розподілені системи використовують мережу, яка також не надає стовідсоткову доступність.

Таким чином, необхідно розробити методологію дослідження високонавантажених програмних додатків яка:

– відповість на існуючі виклики діагностування,

– дозволить моделювання процесів з використанням реальної інформації (як даних, так і операцій),

– буде враховувати систему виконання,

– буде брати до уваги не ідеальність оточуючих частин системи,

– не буде призводити до падіння швидкодії через присутність ефекту наглядача,

– зведе до мінімуму необхідність розміщення оновлення програмного забезпечення з покращеним записом діагностичної інформації.

Використання знімків пам’яті, на думку автора, відповідає існуючим викликам.

Огляд поточних трендів в аналізі пам'яті та оптимізації. Через необхідність глибокої технічної експертизи у розумінні повного циклу виконання програмного забезпечення:

– апаратна частина (виконання процесорних команд, наявність особистої/спільної кеш-пам’яті у процесорних ядер),

– принципів роботи операційної системи (виконання користувача / ядра, наявність гіпервізора, система введення-виведення),

– принципів роботи середовища виконання (трансляції коду, робота з потоками, обробка помилок),

– робота програмного додатку.

Існує критично мало технічно-змістовних публікацій за тематикою. Існуючі програмні рішення здебільшого фокусуються на збиранні та візуалізації діагностичної інформації з логів та метрик, наприклад Azure Application Insights, NewRelic, Datalog. Ці рішення беруть на себе завдання зі збору та агрегації діагностичної інформації. Але саме аналіз та подальші оптимізації є обов’язком інженерів.

Через надвелику площину можливих причин виникнення проблемних ситуацій і їх притаманності саме конкретним системам, використання штучного інтелекту не показує гарних результатів. Ураховуючи значні економічні втрати від проблем зі стабільністю та продуктивністю, передові компанії намагаються вирішити цю проблему через громаду та публічні проекти, такі як ClrMD та PerfView. Вони дозволяють розробникам автоматизувати аналіз знімків пам’яті та профайлів.

Розроблена методологія дослідження неоптимальностей роботи програмного забезпечення. Запропонована методологія дослідження неоптимальностей роботи програмного забезпечення складається з таких кроків:

1. Постановка проблеми, визначення історичної поведінки як неоптимальної, встановлення часового проміжку

2. Застосування USE [6] методології для встановленого часового проміжку:

– Usage – аналіз «використання» для кожної метріки (CPU / RAM / Sockets / Threads / Requests),

– Saturation – чи є «насищення» ресурсів при якому з’являться черги на виконання,

– Errors – помилки, які помилки система записала в журнал.

3. На основі аналізу метрик та визначення проблеми, встановлення автоматичного збору знімків пам’яті.

4. Аналіз знімків пам’яті які містять цілісний процес для виявлення першопричини.

5. Розробка рішення та оцінка його швидкодії відносно базової реалізації з використанням реальних даних і розподілу операцій.

Подальше дослідження спрямовано на знаходження паттернів у знімках пам’яті які є індикаторами неоптимальності виконання програмних додатків.

Висновки. Існуючі системи аналізу неоптимальностей виконання програмних додатків не відповідають викликам сучасних розподілених систем. Відсутність можливостей збору

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

необхідної інформації саме у момент настання збою призводить до створення гіпотез. У подальшому, ці гіпотези можуть бути підтверджені на основі моделей даних які будуть відрізнятись від реальних і не призведуть до вирішення першопричини. Запропонований новий комплексний підхід який дозволяє збирати увесь обсяг даних яким оперує процес, що робить можливим точно ідентифікувати першопричину, розробити покращену модель для її вирішення, та оцінити ефект на повністю відповідних даних.

Сучасні практики проектування програмних додатків будуються на модульності, програмуванні за контрактами, що унеможливлює якісний статичний аналіз коду.

Проблеми стабільності та швидкодії програмного забезпечення постають все частіше при зростанні обсягів інформації, функціоналу, та одночасних користувачів і призводять до прямих матеріальних збитків, зростання загальної вартості володіння (Total cost of ownership).

ЛІТЕРАТУРА

1. Мітіков М.Ю., Гук Н.А. Огляд методів виявлення та аналізу проблем продуктивності в програмному забезпеченні: підходи, виклики та перспективи // Питання прикладної математики і математичного моделювання [Текст]: зб. наук. пр. / редкол.: О.М. Кісельова (відп. ред.) [та ін.]. – Дніпро, 2023. – Вип. 23. – с. 171 – 178. doi: 10.15421/322318

2. Мітіков М.Ю., Гук Н.А. Огляд методів та інструментів системного аналізу продуктивності програмного забезпечення // Математичне та програмне забезпечення інтелектуальних систем (МПЗІС-2023): Тези доповідей XXI Міжнародної науково-практичної конференції, Дніпро, 22-24 листопада 2023 р. / Під загальною редакцією О.М. Кісельової. – Дніпро: ДНУ, 2023. – 213 – 214 с.

3. Мітіков М.Ю., Гук Н.А. Інформаційна технологія діагностики надмірного використання пам’яті на основі аналізу знімків пам’яті // Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості і освіті: Тези XVII Міжнародної науково-практичної конференції (Дніпро, 13-14 грудня 2023 р.). – Д.: УДУНТ, 2023. – с. 32.

4. Louridas P. Static code analysis / P. Louridas. // IEEE Software. – 2006. – Vol. 23, no. 4. – C. 58–61, doi: 10.1109/MS.2006.114.

5. Machine Learning for Anomaly Detection: A Systematic Review / A.B.Nassif, M.A. Talib, Q. Nasir, F.M. Dakalbab. // IEEE Access. – 2021. – Vol. 9. – C. 78658–78700. doi: 10.1109/ACCESS.2021.3083060.

6. Smart detection in Application Insights [Електронний ресурс] / AbbyMSFT, KennedyDenMSFT, AaronMaxwell, v-jbasden // Microsoft. – 2024. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/proactive-diagnostics.

7. Weninger M. Analyzing Data Structure Growth Over Time to Facilitate Memory Leak Detection / M. Weninger, E. Gander, H. Mössenböck. // In Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE \'19). – 2019. – C. 273–284. doi: 10.1145/3297663.3310297.

8. Veasey T. Anomaly Detection in Application Performance Monitoring Data / T. Veasey, S. Dodson. // International Journal of Machine Learning. – 2014. – Vol. 4, no. 2. – C. 120–126. doi: 10.7763/IJMLC.2014.V4.398.

9. Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines / K.Choi, J. Yi, C. Park, S. Yoon. // IEEE. – 2021. – Vol. 9. – C. 120043–120065. doi: 10.1109/ACCESS.2021.3107975.

REFERENCES

1. Mitikov M., Huk N. Review of methods for identifying and analysing performance problems in software: approaches, challenges and prospects // Issues of Applied Mathematics and Mathematical Modelling [Text]: a collection of scientific papers / edited by O.M. Kiseleva (ed.) [and others.] - Dnipro, 2023. - Issue 23. - pp. 171 - 178. doi: 10.15421/322318

2. Mitikov M., Guk N. Review of methods and tools for system analysis of software performance // Mathematical and software of intelligent systems (MPIS-2023): Abstracts of the XXI International Scientific and Practical Conference, Dnipro, 22-24 November 2023 / Edited by O.M. Kiselova - Dnipro: DNU, 2023. 213 - 214 p.

3. Mitikov M., Guk N. Information technology for diagnosing excessive memory usage based on the analysis of memory snapshots // Modern information and communication technologies in transport, industry and education: Theses of the XVII International Scientific and Practical Conference (Dnipro, 13-14 December 2023): USUNT, 2023. - p. 32.

4. Louridas P. Static code analysis / P. Louridas. // IEEE Software. – 2006. – Vol. 23, no. 4. – C. 58–61, doi: 10.1109/MS.2006.114.

5. Machine Learning for Anomaly Detection: A Systematic Review / A.B.Nassif, M.A. Talib, Q. Nasir, F.M. Dakalbab. // IEEE Access. – 2021. – Vol. 9. – C. 78658–78700. doi: 10.1109/ACCESS.2021.3083060.

6. Smart detection in Application Insights [Електронний ресурс] / AbbyMSFT, KennedyDenMSFT, AaronMaxwell, v-jbasden // Microsoft. – 2024. – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/proactive-diagnostics.

7. Weninger M. Analyzing Data Structure Growth Over Time to Facilitate Memory Leak Detection / M. Weninger, E. Gander, H. Mössenböck. // In Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE \'19). – 2019. – C. 273–284. doi: 10.1145/3297663.3310297.

8. Veasey T. Anomaly Detection in Application Performance Monitoring Data / T. Veasey, S. Dodson. // International Journal of Machine Learning. – 2014. – Vol. 4, no. 2. – C. 120–126. doi: 10.7763/IJMLC.2014.V4.398.

9. Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines / K.Choi, J. Yi, C. Park, S. Yoon. // IEEE. – 2021. – Vol. 9. – C. 120043–120065. doi: 10.1109/ACCESS.2021.3107975.

Received 24.06.2024.
Accepted 27.06.2024.

Modern problems of anomaly identification in Enterprise Systems

The article addresses modern challenges in anomaly detection within enterprise systems using memory dump analysis. As the complexity of enterprise systems grows, the number of potential issues affecting their stability and performance also increases. Anomalies, such as software failures or unexpected deviations from normal behavior, can lead to serious consequences, including data

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

loss, reduced performance, or even complete system shutdown. Detecting and resolving these anomalies is a critical task for maintaining uninterrupted operation in enterprise environments.

The primary method discussed in this article is memory dump analysis, which provides detailed information about the system's state at the time of an anomaly. This method is effective for identifying root causes of failures, such as memory leaks or other resource-intensive operations. However, due to the large volumes of data and the complexity of modern software systems, memory dump analysis faces several challenges, such as the need for precise data collection during incidents and the requirement for powerful computational resources to process such data.

The article thoroughly analyzes algorithms and tools used for detecting problems in enterprise systems. Specifically, statistical methods, machine learning algorithms, and tools for memory dump analysis are reviewed. Machine learning techniques enable the creation of models representing normal system behavior and automatically detect deviations from these models, facilitating timely identification of potential issues. Additionally, optimization methods aimed at improving system performance, including techniques such as parallelization, caching, and code profiling, are explored.

One of the main challenges discussed in the article is the limitations of existing methods and tools for software analysis. High-load systems often face difficulties in real-time profiling and monitoring, complicating the identification of root causes. The article also examines limitations related to the accuracy of data collection and the complexity of diagnosing issues in distributed systems.

Based on the analysis, the article suggests future prospects for improving modern methods of anomaly detection in enterprise systems. Key areas for further research include enhancing machine learning algorithms for memory dump analysis, developing more efficient optimization methods, and improving monitoring tools to increase the accuracy and speed of problem detection. The article also highlights the importance of integrating these technologies into real-world enterprise environments to ensure stability and reliability.

Keywords: anomaly detection, enterprise systems, memory dumps, performance optimization, problem detection, machine learning, code profiling, optimization algorithms, system monitoring, distributed systems.

Гук Наталія Анатоліївна – завідувачка кафедри комп'ютерних технологій, доктор фізико-математичних наук, професорка, Дніпровський національний університет імені Олеся Гончара, huk_n@fpm.dnu.edu.ua, ORCID ID: 0000-0001-7937-1039.

Мітіков Микола Юрійович – аспірант кафедри прикладної математики, Дніпровський національний університет імені Олеся Гончара, mitikov.m22@fpm.dnu.edu.ua, ORCID ID: 0009-0002-1297-5676.

Huk Nataliia - Head of the Department of Computer Technologies, Doctor of Physical and Mathematical Sciences, Professor, Oles Honchar Dnipro National University, huk_n@fpm.dnu.edu.ua, ORCID ID: 0000-0001-7937-1039.

Mitikov Nikolay - PhD student, Department of Applied Mathematics, Oles Honchar Dnipro National University, mitikov.m22@fpm.dnu.edu.ua, ORCID ID: 0009-0002-1297-5676.