УДК 004

ТЕХНОЛОГІЇ САМОВІДНОВЛЕННЯ КОМПОНЕНТІВ ОПЕРАЦІЙНОЇ СИСТЕМИ.

О. В. Фокін[1,2, a]

[1] *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Фізико-технічний інститут*

[2] *Samsung R&D Institute Ukraine (SRK)*

Анотація

Розглянуто актуальну на сьогодні проблему моніторингу та відновлення операційної системи та існуючі підходи, що можуть забезпечити критерій доступності. Розглянуто переваги та недоліки підходів: TTL, перевірка доступності, реактивного самовідновлення, профілактичного самовідновлення. На базі аналізу розроблена модель моніторингу та відновлення операційної системи, яка основана на штучному інтелекті.

Ключові слова: системи самовідновлення, системи моніторингу, операційні системи, штучний інтелект

Вступ

В даний час вимоги до операційних систем стають усе деталі жорсткіші, а кількість компонентів системи зростає. Одною з основних вимог до таких систем є забезпечення ними критерію доступності, тому все більше виникає необхідність у системах автоматичного моніторингу та самовідновлення компонентів ОС, а також оповіщення користувача та розробника о виникненні несправності при роботі компоненту[1, 2]. У даній роботі ми пропонуємо модуль для виявлення аномальної поведінки компонентів ОС та алгоритму дій для відновлення системи в нормальний стан.

1. Існуючі рішення

На сьогодні існує багато технологій для моніторингу та самовідновлення компонентів операційної системи шляхом прийняття рішень при відмові одного з них. Такі технології застосовуються для забезпечення доступності операційної системи. Технології самовідновлення можуть бути реалізовані на рівні застосунків, системи або апаратному рівнях.[1, 2, 3, 4] Для моніторингу компонентів операційної системи можуть використовуватися такі підходи, а саме:

1) Час життя (TTL) [1, 3]
2) Перевірка доступності [1]

А для самовідновлення компонентів можно виділити наступні підходи:

1) Реактивне самовідновлення [1]
2) Профілактичне самовідновлення[1]

1.1. Підхід моніторингу на основі часу життя (TTL)

Перевірки часу життя (TTL) припускають, що служба або додаток будуть періодично підтверджувати свою працездатність. Система, яка приймає сигнали TTL, відстежує останній повідомлений стан для даного TTL. Якщо цей стан не оновлюється протягом попередньо визначеного періоду, система моніторингу передбачає, що служба не виконана і її необхідно відновити до початкового стану. Якщо процес, в якому працює служба, завершиться невдало, він не зможе відправити запит, закінчиться TTL і будуть виконані заходи реагування.[1, 3]

Основна проблема TTL – це те, що програми та сервіси повинні бути прив'язані до системи моніторингу. Впровадження TTL було б одним з антипатернів мікросервісів, бо ми намагаємося спроектувати їх так, щоб вони були максимально автономними. Більш того, мікросервіси повинні мати чітку функцію і єдину мету. Реалізація запитів TTL всередині них додасть додаткову функціональність і ускладнить розробку.

1.2. Підхід моніторингу на основі перевірки доступності

Підхід на основі пінгування полягає в тому, щоб перевіряти зовнішній стан програми або служби. Система моніторингу повинна періодично перевіряти кожну послугу і якщо відповідь не отримана або зміст відповіді не є коректним, виконати заходи з відновлення.[1]

Пінгування протилежне до TTL і по можливості найбільш прийнятний шлях перевірки стану окремих частин системи.

1.3. Підхід відновлення системи на основі реактивного самовідновлення

Підхід на основі реактивного самовідновлення полягає в тому, щоб після того як засоби моніторингу

[a] pandorus2012@gmail.com

зафіксували неправильну поведінку компоненту системи або зупинення його роботи через непередбачену помилку, система відновлює свій стан до заданого. Доки у нас є всі перевірки, а також дії, які повинні бути виконані в разі збою, ми зводимо час простою системи до мінімуму.[1]

1.4. Підхід відновлення системи на основі профілактичного самовідновлення

Ідея профілактичного лікування полягає в тому, щоб передбачити проблеми, які можуть виникнути в майбутньому, і діяти таким чином, щоб уникнути цих проблем. Простий, але менш надійний спосіб прогнозування майбутнього полягає в тому, щоб засновувати припущення на даних в реальному часі. При такому підході аналізується час відгуку системи, завантаженість процесора, пам'яті та інше. [1] Наприклад, ми могли б помітити, що протягом останньої години використання пам'яті неухильно зростало і досягло критичного рівня, скажімо, 90%. Це було б чіткою ознакою того, що послуга, що викликає таке збільшення, повинна маштабуватися. Система також може враховувати більш тривалий період часу і робити висновок про те, що щопонеділка відбувається раптове збільшення трафіку, і завчасно масштабувати послуги, щоб запобігти тривалої відповіді.[1]

2. Постановка проблеми

У цій роботі розглядаються переваги та недоліки різних підходів для моніторингу та відновлення компонентів ОС та на базі цього аналізу розробка власної моделі. Так наприклад підхід моніторингу TTL ускладнює розробку сервісів і це пов'язано з тим, що необхідно реалізовувати TTL всередині самих компонентів ОС. Цю проблему вирішує підхід на базі пінгування, але його не завжди можна реалізувати у зв'язку з архітектурними особливостями деяких компонентів. Аналізуючи усі підходи можна зробити висновок о необхідності розробки системи, яка б була гнучка, тобто могла аналізувати з яким типом компоненту вона зараз працює та який оптимальний підхід обрати для моніторингу та відновлення у разі збоїв.

Об'єктом дослідження є технології самовідновлення компонентів ОС.

Предметом дослідження є методи моніторингу компонентів ОС, та завдання забезпечення доступності, в яких технології самовідновлення беруть учать.

2.1. Використання AI у моніторингу та відновленні ОС

Використання AI для моніторингу та відновлення системи дуже важливо. На сьогодні існує багато різноманітних сервісів, які відрізняються не тільки задачами, які вони виконують, але і своєю архітектурою. Неправильно застосовувати одні та ті ж методи та параметри для різних сервісів ОС. Система моніторингу повинна буду гнучкою та аналізуючи усі данні підбирати оптимальні параметри роботи усіх своїх компонентів. Саме використання AI може розв'язувати цю проблему контролюючи роботу системи, будуючи портрети сервісів на основі даних з модулів моніторингу та зберігання їх у базі даних, фіксувати неправильну роботу та приймати рішення з відновлення ОС.

3. Загальна архітектура

Нашою метою є аналіз та розробка автоматизованої системи для самовідновлення компонентів ОС, яка може автоматично виявити некоректність у роботі одного з компонентів ОС та вжити заходів для забезпечення доступності ОС. Цього слід досягти шляхом поєднання усіх підходів, які існують у даний час та така система не повинна сильно впливати на продуктивність ОС. Така системи повинна аналізувати сервіс з яким вона працює і на базі отриманих даних вибирати метод моніторингу та відновлення.

3.1. Деталі підходу

На базі усіх переваг та недоліків різних підходів можна пободувати нову модель системи моніторингу та відновлення ОС. Після того як у системі запускається новий компонент, запропонована модуль працює по наступному алгоритму на Рисунку1:

1) Модуль «Resource monitoring» повідомляє модулю «General function», що у системі почав працювати новий компонент.
2) «General function» відправляє запит до компонента з метою отримати доступні підходи для моніторингу і в залежності від відповіді підключає TTL модуль, або модуль пінгування. Модуль «Resource monitoring» підключається завжди, незалежно від відповіді компоненту системи.
3) Після підключення модулів моніторингу, «General function» відправляє данні до модуля AI, який у свою чергу повертає параметри моніторингу та починає обробляти данні з систем мониторингу. У разі фіксування неправильної поведінки компоненту, він відправляє команду до «General function» на відновлення системи.
4) «General function» починає обробляти команду, починаючи з параметр «Ignore». Далі модуль перевіряє значення параметру «Critical» і якщо він буде дорівнювати значенню «true», система негайно виконає процедуру «kill process». У випадку коли параметр «Critical» дорівнює значенню «false» модуль відправляє повідомлення користувачу системи для прийняття рішення.
5) Користувач у свою чергу може обрати закриття сервісу, або ігнорування. Якщо було обрано закриття сервісу, параметр «Critical» прирівнюється до «true». При ігноруванні параметр «Ignore» прирівнюється до «true». Вибір відправляється до модуля «Resource monitoring» і після цього ми повертаємося до пункту 4.

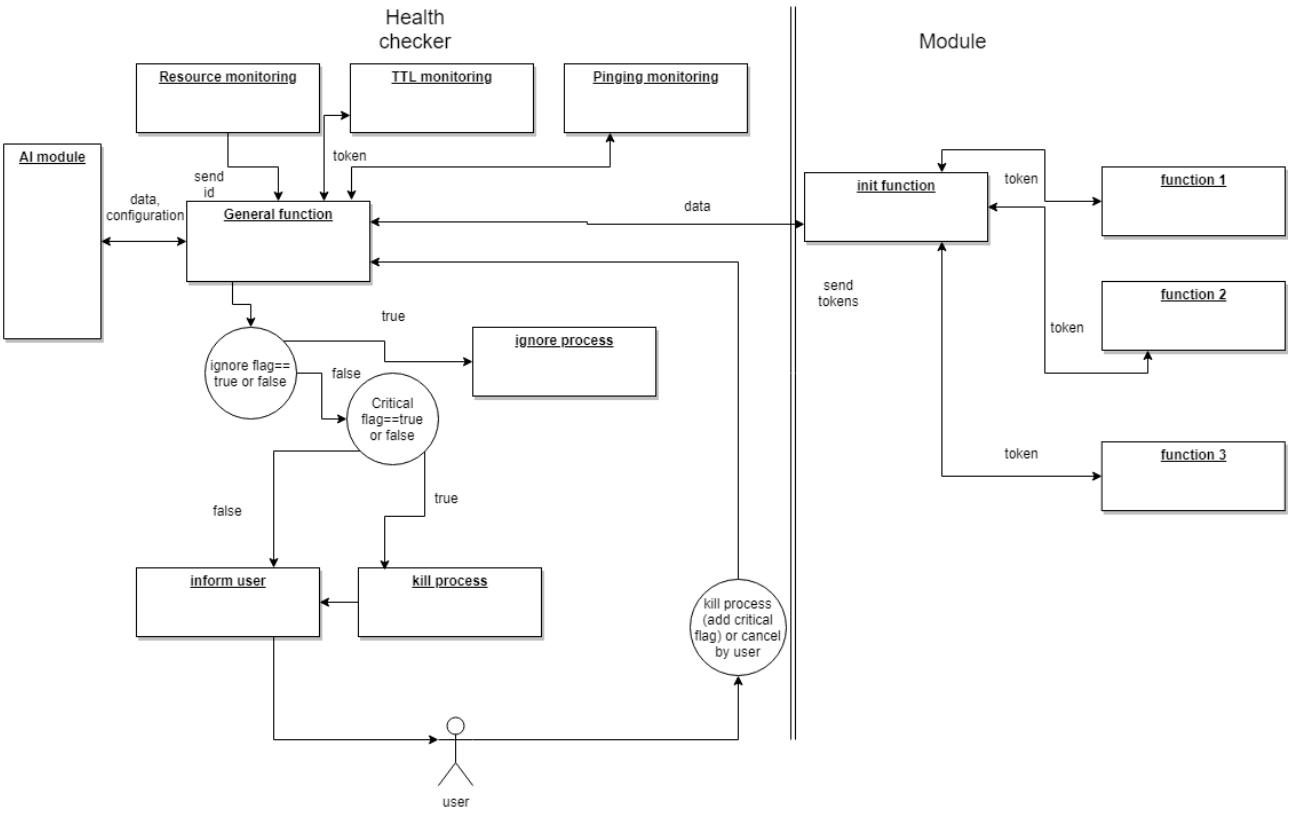

Рис. 1. Схема запропонованої моделі моніторингу та відновлення ОС

Висновки

Розглянуто різноманітні підходи моніторингу та відновлення операційної системи у разі неправильної роботи одного з її компонентів, а також переваги та недоліки цих підходів.

Проведено аналіз та розроблена модель на базі цих підходів з використанням штучного інтелекту. Дана концепція стала розширенням модулю самовідновлення операційної системи Android у Bluetooth.

В майбутньому, буде реалізована запропонована модель та проаналізовано вплив такої системи на продуктивність ОС, а також ефективність системи при розв'язання питання доступності ОС.

Перелік використаних джерел

1. Farcic Viktor. The Devops 2.0 Toolkit. — Packt Publishing Limited, 2016. — ISBN: 1785289195. — Access mode: `https://www.amazon.com/Devops-2-0-Toolkit-Viktor-Farcic/dp/1785289195?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=1785289195`.
2. Saha Goutam. Software - Implemented Self-healing System // CLEI Electron. J. — 2007. — 12. — Vol. 10.
3. Chaudhry Junaid Ahsenali. Self-Healing Systems and Wireless Networks Management. — CRC Press, 2013. — oct. — Access mode: `https://www.xarg.org/ref/a/B00FOVFX98/`.
4. Koopman Philip. Elements of the Self-Healing System Problem Space. — 2003. — 01.