УДК 004.7:004.415-025.25 **doi: 10.26906/SUNZ.2022.1.048**

М. О. Волк, О. Г. Лунічкін

Харківський національний університет радіоелектроніки, Харків, Україна

КОМП'ЮТЕРНІ СИСТЕМИ ІЗ САМОВІДНОВЛЕННЯМ

Анотація. Одною з властивостей сучасних комп'ютерних систем є забезпечення їх надійної безперервної роботи, що реалізується різними засобами, у тому числі й вбудованими у саму систему. **Предметом** статті є огляд проблеми самовідновлення в комп'ютерних системах. **Мета статті** полягає у створенні класифікаційної бази щодо ймовірних відмов, процесів зберігання стану компонентів системи та задач забезпечення самовідновлення комп'ютерних систем. **Висновки.** Проаналізовано тематичну літературу предметної області. Запропонована класифікація відмов, описані види відновлення із застосуванням бекапів, сформульовані перспективні задачі управління комп'ютерними системами із самовідновленням. Незважаючи на існуючі приклади реалізації принципів самовідновлення у комп'ютерних системах, самовідновлення у вигляді окремих сервісів чи компонентів є новим напрямком досліджень. Результати можуть набути чималого поширення та застосовуватись у розробці більшості програм та пристроїв, інформаційних систем з метою спрощення проектування, налагодження та експлуатації сучасних комп'ютерних систем.

Ключові слова: комп'ютерна система, самовідновлення, відмова, бекап.

Вступ

Сучасні комп'ютерні сервіси не лише стали невід'ємною частиною виробництва та наукової діяльності, але й набули чималого поширення серед повсякденного життя кожної людини. Спілкування, робота, електронні документи, онлайн покупки, банкові операції – велика частка нашої побутової діяльності покладена на комп'ютери та програми. Тож зі зростанням кількості таких сервісів та додатків зростає й складність та відповідальність забезпечення неперервної діяльності таких систем, а саме – стійкості до відмов. Постає актуальною задача з максимальної автоматизації усіх складових забезпечення відмовостійкості – від прогнозування до усунення наслідків відмови.

Аналіз публікацій

Теорія забезпечення відмовостійкості існує з моменту появи комп'ютерних систем. Проте реалізація відмовостійкості як внутрішньої властивості самої системи, що реалізується окремим сервісом або компонентом системи, є новим напрямом наукових досліджень. Потреба у виокремленні відмовостійкості та самовідновлення як окремих напрямків виникла у зв'язку з надзвичайним поширенням електронних сервісів у побуті. Чимала кількість додатків та відповідного серверного забезпечення потребують великої кількості часу та працівників для підтримки, тому фахівці почали приділяти окрему увагу оптимізації та автоматизації попередження та усунення наслідків відмов.

Саме на виокремлення напряму досліджень та формування термінології спрямована робота автор роботи [1]. Закінчує роботу автор формулюванням тем для подальшого розвитку систем з самовідновленням.

У роботі [2] поглиблюється питання термінології, зокрема наведено приклади алгоритмів самовідновлення у природі. Автори створюють загальну схему процесу самовідновлення та наводять декілька можливих стратегій для реалізації такої властивості системи.

Задля швидкого та ефективного створення програм та апаратних комплексів із самовідновленням необхідні відповідні архітектурні рішення. Тож у роботі [3] формалізовано архітектуру системи із самовідновленням. Для створення додатків пропонується архітектура на основі подій, а для формалізованого опису системи пропонується окреме середовище, яке також спроможне формалізувати різницю між станами системи, на основі яких створюється план відновлення.

Практичне застосування принципів самовідновлення описано у роботі [4]. Автор описує реалізацію самовідновлення у системах, які побудовані на компонентній архітектурі OpenORB. Наведено три приклади застосування такої архітектури у існуючих програмних системах.

Відмовостійкість та самовідновлення є властивостями систем, що подовжують час експлуатації та спрощують обслуговування та підтримку, тому розробники зацікавлені у реалізації таких властивостей у програмах чи пристроях. Із еволюцією програмних технологій та засобів розробки з'являються інструменти, які дозволяють реалізовувати вказані принципи на рівні архітектури, фреймворків і навіть мережних протоколів [5-7]. Тож перед дослідниками постає проблема відокремлення та узагальнення моделі та методів самовідновлення. Зокрема, цим питанням займаються автори робот [1, 2]. Теоретичні дослідження доцільно супроводити практичними розробками, як пропонує автор роботи [3], який описує архітектуру системи самовідновлення. Автор [4] описує програмну імплементацію принципів самовідновлення.

З недоліків оглянутої літератури варто зазначити відсутність формалізованого визначення відмови у комп'ютерній системі та не розглянуті механізми забезпечення процесів виявлення відмов та відновлення.

Постановка проблеми

Перед побудовою системи з самовідновленням необхідно формалізувати відмови та відокремити нештатні ситуації, які вважатимуться відмовами та потребуватимуть процесу відновлення. Також необхідно формалізувати різні етапи відновлення, які у подальшому можуть стати предметом для автоматизації, та, відповідно, основою сервісу або компонентами підсистеми самовідновлення [8-13]. Маючи перелік відмов

та етапів усунення наслідків доцільно запропонувати перспективні напрямки роботи щодо можливої автоматизації та побудови систем із самовідновленням.

Тож метою роботи є створення класифікаційної бази щодо ймовірних відмов, процесів зберігання стану компонентів системи та задач забезпечення самовідновлення комп'ютерних систем.

Головні причини відмов

1. Помилка у програмі під час виконання. Причиною є помилка у написанні програми або неправильні вхідні дані. Виявляється програмою та генерується виключення, або завершується виконання програми. Для запобігання таких помилок достатньо існуючого механізму виключень.

2. Помилка на рівні операційної системи. Прикладами таких помилок можуть бути:

- читання/запис по некоректному адресу пам'яті,
- ділення на нуль,
- нестача пам'яті та ін.

Операційна система дозволяє обробити ці помилки аналогічно виключенням, але цілісність даних може бути порушена, тому у деяких випадках стан програми доцільно вважати невиправним та зупиняти виконання. Необхідно застосування окремого механізму відновлення.

3. Пошкодження операційної системи або апаратури. Тобто помилки, які порушують працездатність техніки за незалежних від програми причин. Відновлення роботи програми на ушкодженому комп'ютері неможливо. Для продовження роботи необхідно мати розподілену систему та переводити користувача до іншої машини з максимальним збереженням даних з мінімальним часом відновлення.

Види бекапів для самовідновлення

Окрім програмних виключень, інші види відмов призводять до втрати екземпляра програми. Якщо активними є декілька екземплярів однієї програми, то при відмові одного залишаються інші, здатні ще деякий час виконувати покладені задачі, поки готується новий екземпляр на зміну ушкодженому. Але, у найгіршому випадку, якщо екземпляр програми лише один, задачею самовідновлення є перезапуск програми із відновленням стану за найкоротший час. Це означає, що стан програми потрібно зберегти до (або під час) відмови (тобто зробити бекап – резервне копіювання даних). У випадку фізичного ушкодження апаратури, яке неможливо передбачити, повністю зберегти актуальний стан програми на момент відмови неможливо. Тому для можливості будь-якого відновлення необхідно заздалегідь зберігати стан програми у іншому місці. За місцем зберігання стану можна поділити механізми самовідновлення на чотири групи.

1. Локальний бекап. Стан програми зберігається на жорсткому диску в комп'ютері, на якому виконується програма. Не захищає від ушкоджень комп'ютера, проте захищає від програмних помилок та некоректних дій користувача у програмі, що призводять лише до зупинки виконання програми. Найпростіший метод зберігання стану, активно використовується у сучасних застосунках. Для повноцінного самовідновлення необхідно додати сервіс, який буде автоматично запускати програму після відмови та повідомляти їй останній бекап.

2. Виділений бекап сервер. Стан програми відсилається на сервер, який умовно вважається невідмовним (тобто, за допомогою адресу цього сервера завжди можна отримати стан програми). Для відновлення роботи системи диспетчеру необхідно обрати новий доступний комп'ютер, завантажити бекап із серверу та запустити програму з останнім збереженим станом.

3. Розподілена однорівнева система. Умовно невідмовного серверу немає, і усі пристрої під загрозою відмови. Тому, в процесі надсилання стану програми, існує ймовірність його втрати водночас з пристроєм або каналом зв'язку. Для відновлення необхідно шукати бекап у функціонуючих пристроях, та запустити програму на новому пристрої. Найскладніша архітектура системи для самовідновлення, адже необхідно якнайширше відсилати стан програми, аби мінімізувати риск повної втрати даних.

4. Розподілена дворівнева система. На першому (фізичному) рівні є сукупністю розподілених пристроїв аналогічних третьої групи. Відрізняється наявністю другого (хмарного) рівня, який забезпечує надійне зберігання станів програм на віддалених серверах. Другий рівень бере на себе реалізацію гарантованої надійності зберігання даних [14]. Виділення двох рівнів спрошує реалізацію відновлення станів програмних компонентів системи та надає можливість застосовувати існуючі хмарні сервіси збереження даних.

За періодичністю зберігання стану можна поділити механізми самовідновлення на три групи:

1. Нерегулярні бекапи. Стан програми зберігається лише у окремих випадках, наприклад, введені важливі дані у інтерфейсі користувача, механізм прогнозування відмов просигналізував про можливість відмови (наприклад, оперативна пам'ять майже закінчилась), користувач натиснув відповідну кнопку та ін. Найлегший у реалізації вид бекапів у реалізації, адже не потрібна оптимізація процесів отримання, відправки та зберігання даних.

2. Регулярні бекапи. Стан програми зберігається з деякою періодичністю. Зазвичай, період береться найменший, який середовище виконання дозволяє роботи. Першим обмеженням періоду бекапу є працездатність комп'ютера – постійне копіювання великого обсягу даних на жорсткий диск знижує продуктивність комп'ютера. Наступним обмеженням є пропускна здатність мережі – мережеве обладнання має пропускну здатність, оператор за тарифом також надає обмежену пропускну здатність. Окрім того, є трафік, необхідний іншим сервісам та додаткам. Тому посилати у мережу бекапи занадто часто також не ефективно.

3. Бекап у реальному часі. Кожна операція, що змінює стан програми, зберігається або відсилається. Таким чином стан програми у бекапі завжди актуальний, та можливо повне відновлення без втрати даних. Але це значно збільшує час виконання програми, адже кожен програмний крок необхідно зберігати або відправляти. Тому такі системи потребують пристрій зберігання з малим часом запису або надійне поєднання зі іншими комп'ютерами у мережі. Також існують меха-

нізми зберігання не усіх даних програми, а історії зміни окремих даних (спираючись на факт, що усі данні неможливо водночас змінити) [15].

Таким чином, механізм самовідновлення у загальному випадку працює наступним чином: програма під час виконання зберігає свій стан, у випадку відмови припиняє виконання, далі програма запускається заново, їй надається останній стан, програма продовжує своє виконання. Конкретні алгоритми відрізняються місцем зберіганням станів та періодичністю бекапів.

Чим більше різних пристроїв зберігає стан програми, тим довше та складніше оновлювати його, але тим менше риск повної втрати даних (якщо немає виділеного бекап серверу), та потенційно менший час відновлення (адже можна з декількох пристроїв обрати найближчий, з якого швидше завантажиться стан).

Чим менше період бекапів, тим менше даних втрачаються у разі відмови. У разі зберігання стану у реальному часі дані не втрачаються взагалі. У невеликих системах із невеликим об'ємом даних бекап у реальному часі досить нескладно зробити. Наприклад, у хмарному застосуванні редагування текстових документів Google documents кожна маніпуляція з тексом одразу потрапляє у хмарне сховище, тому будь-які відмови на клієнтському пристрої не пошкодять дані, а відкрити той самий документ можливо швидко на іншому пристрої. Але у випадку завантажених сервісів, такі як соціальні мережі, потік нових даних може бути занадто великий для негайного зберігання у декількох сховищах, тому розробникам та адміністраторам необхідно підбирати баланс між частотою бекапів, аби і клієнт не втратив занадто багато своїх повідомлень/фотографій/відеозаписів, і датацентри не були постійно стовідсотково зайняті копіюванням існуючих даних.

Структурні елементи вирішення проблем самовідновлення

Для реалізації ефективного самовідновлення система повинна мати відповідні компоненти та властивості. Наведемо елементи, які зазвичай можуть не реалізуватися у традиційних програмних або апаратних комплексах, не здатних до самовідновлення.

1. **Спостерігач**. У системі необхідний програмний компонент або пристрій, який буде займатися моніторингом інших програм та пристроїв комп'ютерної системи, та у разі відмови ініціювати відновлення і передавати останній бекап. Спостерігач повинен завжди бути активним, тож у одноріневих системах, коли усі пристрої однаково під загрозою відмови, постає питання, скільки пристроїв мають бути спостерігачами, як передаватимуться повноваження спостерігача після його відмови, та яку архітектуру матиме така система.

2. **Прогнозування**. Прогнозування відмови дозволятиме готувати актуальний бекап та мінімізувати втрату даних. Але у загальному випадку прогнозувати відмову неможливо, тож алгоритми розробляються для конкретних випадків (синоптики прогнозують маршрут тайфуну, радарні системи будують траєкторію ворожої ракети, операційна система фіксує аномальне зростання пам'яті процесу, тощо).

3. **Діагностика** причин відмови. З'ясування факторів, що призвели до відмови. Спрощує попередження аналогічних відмов. У програмних системах часто недооцінена проблема, адже після відмови програми у більшості випадків файл із подробицями містить хаотичну інформацію, яка не допоможе розробнику знайти причину відмови.

4. **Самовідновлення** у реальному часі. Інколи, наприклад, у системах відеоспостереження, відновлення повинно відбуватися якомога швидше, тому оптимізувати необхідно і завантаження бекапу, і час старту програми, або взагалі, програма вже повинна бути запущена заздалегідь до відмови.

5. **Корегування**. Після відмови доречно виправити помилки, що призвели до відмови, та підготувати систему до наступної або запобігти аналогічним відмовам. Традиційні програми не можуть самі себе виправити та перезібрати, проте існують напрямки реалізації цих можливостей [16].

6. **Виявлення порушення**. У деяких випадках, наприклад, втручання злочинця у роботу програми, застосування продовжує роботу без відмови. Щоби виявити втручання недостатньо чекати виключень чи сигналів операційної системи, а потрібно активно перевіряти дані додатку та дії користувачів. Якщо злочинець працює акуратно та з мінімальним втручанням, помітити втручання складно, що може потребувати нетривіальних алгоритмів виявлення таких подій (наприклад, засобами машинного навчання).

На основі наведених структурних елементів, які пропонуються реалізовувати у комп'ютерних системах, можливе створення шаблонів програмування [17, 18], за якими будуть реалізовуватися програмні об'єкти підтримки процесів самовідновлення.

Висновки

Самовідновлення є корисною властивістю, необхідною для побудови надійних та простих у супроводі програмних та апаратних систем. Розробники зараз застосовують алгоритми самовідновлення для вирішення деяких проблем для збільшення надійності комп'ютерних систем. Актуальним напрямом наукової праці є формалізований опис процесів відмови та виправлення наслідків відмови окремих компонентів системи, розробка узагальненої моделі та методів самовідновлення, які можна застосувати для більшості програмних та апаратних розробок. Перспективним напрямом практичної діяльності є створення архітектури та її імплементація у сервісі самовідновлення або компоненті, що вирішуватиме задачу самовідновлення.

В роботі оглянута тематична література предметної області самовідновлення в комп'ютерних системах. Класифіковані типові відмови у комп'ютерних системах, класифіковані деякі методи самовідновлення, описані види відновлення із застосуванням бекапів, сформульовані перспективні задачі управління комп'ютерними системами із самовідновленням. Незважаючи на існуючі приклади реаліза-

ції принципів самовідновлення у комп'ютерних системах, самовідновлення у вигляді окремих сервісів чи компонентів є новим напрямком досліджень. Результати можуть набути чималого поширення та застосовуватись у розробці більшості програм та пристроїв, інформаційних систем з метою спрощення проектування, налагодження та експлуатації сучасних комп'ютерних систем.

СПИСОК ЛІТЕРАТУРИ

1. Rodosek, Gabi & Geihs, Kurt & Schmeck, Hartmut & Stiller, Burkhard. (2009). Self-Healing Systems: Foundations and Challenges. Self-healing systems — survey and synthesis
2. Ghosh, Debanjan & Sharman, Raj & Rao, Raghav & Upadhyaya, Shambhu. (2007). Self-healing systems — survey and synthesis. Decision Support Systems. 42. 2164-2185. 10.1016/j.dss.2006.06.011.
3. Dashofy, Eric & van der Hoek, Andre & Taylor, Richard. (2002). Towards Architecture-based Self-Healing Systems. Proceedings of the first ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02). 21-26. 10.1145/582128.582133.
4. Blair, Gordon & Coulson, Geoff & Blair, Lynne & Duran-Limon, Hector & Grace, Paul & Silva Moreira, Rui & Parlavantzas, Nikos. (2002). Reflection, Self-Awareness and Self-Healing in OpenORB. Proceedings of the first ACM SIGSOFT Workshop on Self-Healing Systems (WOSS'02). 9-14. 10.1145/582128.582131.
5. Nikolic, J., Jubatyrov, N., Pournaras, E. Self-Healing Dilemmas in Distributed Systems: Fault Correction vs. Fault Tolerance. IEEE Transactions on Network and Service Management, 2021, 18(3),9466159, с. 2728-2741
6. Cai, S., Xie, Y., Zou, Y., Wu, H., Shen, L. A Consensus-based Decentralized Algorithm for Service Restoration in Active Distribution Networks. Chinese Control Conference, CCC, 2021-July, с. 6778-6783
7. Pozo, F., Rodriguez-Navas, G., Hansson, H. Self-Healing Protocol: Repairing Schedules Online after Link Failures in Time-Triggered Networks. Proceedings - 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2021, с. 129-140
8. Chu, T., Wang, T., Cao, C., Huang, W., Wang, Y. Self-healing Control Method in Abnormal State of Distribution Network. Proceedings - 2020 Chinese Automation Congress, CAC 2020, 9326736, с. 6438-6443
9. Shefer O.V., Alnaeri Frhat Ali. Optimum flow distribution in the network with adaptive data transfer. *Electronics and Control Systems*. 2020. No. 4(66). P.45-50. DOI: https://doi.org/10.18372/1990-5548.66.15254
10. Кучук, Н., Шефер, О., Чернева, Г., & Алнаері, Ф. А. (2021). Визначення пропускних здатностей самовідновлювального сегмента мережі. *Сучасні інформаційні системи*, *5*(2), 114–119. https://doi.org/10.20998/2522-9052.2021.2.16
11. Fang, Shuguang, Dong, Yuning and Shi, Haixian (2012), "Approximate Modeling of Wireless Channel Based on Service Process Burstiness", Proceedings of the International Conference on Wireless Networks (ICWN); Athens: 1-7. Athens: The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing, (WorldComp).
12. Sobieraj, M., Stasiak, M. and Weissenberg, J. (2012), "Analytical model of the single threshold mechanism with hysteresis for multi-service networks", IEICE Transactions on Communications, Vol. E95.B No. 1, pp. 120–132.
13. Kuchuk, G., Kovalenko, A., Komari, I.E., Svyrydov, A. and Kharchenko, V. (2019), "Improving Big Data Centers Energy Efficiency: Traffic Based Model and Method", In: Kharchenko V., Kondratenko Y., Kacprzyk J. (eds) Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control, vol 171. Springer, Cham, DOI: https://doi.org/10.1007/978-3-030-00253-4_8.
14. Lynch, J., Ashok Joshi, D. Towards Practical Self-Healing Distributed Databases. IEEE Infrastructure Conference 2020, IEEE Infra 2020, 9377621
15. Волк М.А. Журнализация состояний программных распределенных моделей и ее использование в оптимистических алгоритмах синхронизации. Збірник наукових праць Харківського університету Повітряних Сил. 2010, випуск 1 (23). С.104–107.
16. Рубан І.В., Волк М.О., Рісухін М.В. Метод самовідновлення розподіленого програмного забезпечення в гетерогенних комп'ютерних системах. Телекомунікаційні та інформаційні технології. 2019. № 3 (64), с. 17-23
17. Dias, J.P., Sousa, T.B., Restivo, A., Ferreira, H.S. A Pattern-Language for Self-Healing Internet-of-Things Systems. Proceedings of the European Conference on Pattern Languages of Programs 2020July 2020 Article No.: 25, Pages 1–17, https://doi.org/10.1145/3424771.3424804
18. Fardin Abdi, Rohan Tabish, Matthias Rungger, Majid Zamani, and Marco Caccamo. 2017. Application and system-level software fault tolerance through full system restarts. In 2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS). IEEE, 197--206.

Received (Надійшла) 11.01.2022
Accepted for publication (Прийнята до друку) 23.03.2022

Self-healing computer systems

Maksym Volk, Oleksii Lunichkin

Abstract. One of the properties of modern computer systems is providing robust uninterrupted worktime, which can be implemented in different ways, including system internal solutions. Article **subject** is computer system self-healing problem overview. The **goal** of this article is creating classification baseline for potential faults, backup processes and self-healing challenges in computer systems. **Results**. Subject sources were analyzed. Proposed fault classification, described different ways of restoring systems using backups, defined potential challenges in controlling self-healing computer systems. Despite the existing examples of the implementation of the principles of self-healing in computer systems, self-healing in the form of individual services or components is a new area of research. The results can be widely disseminated and used in the development of most programs and devices, information systems to simplify the design, commissioning and operation of modern computer systems.

Keywords: computer system, self-healing, fault, backup.