РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ НАДАННЯ ТА УПРАВЛІННЯ VPS-СЕРВЕРАМИ

Будзінський Артем Олександрович
Здобувач вищої освіти бакалаврського рівня
Дудник Олександр Олександрович
Кандидат технічних наук, доцент
Кафедра програмного забезпечення
Вінницький національний технічний університет, Україна

Анотація

Досліджено проблему централізованого моніторингу та розподілу ресурсів у межах віртуальних виділених серверів (VPS). Проаналізовано наявні платформи віртуалізації та визначено їхні обмеження при роботі на невеликих локальних кластерах. Запропоновано сервісно-агентну архітектуру та текстову логіку визначення загального навантаження обчислювальних вузлів. Реалізовано програмний комплекс, здатний в автоматизованому режимі масштабувати ресурси, запускати резервні сервери та відновлювати роботу системи після апаратних збоїв.

Ключові слова: VPS, віртуалізація, розподілені системи, моніторинг ресурсів, автоматизоване масштабування, gRPC, сервісно-агентна архітектура.

Abstract

The paper investigates the problem of centralized monitoring and resource allocation within virtual dedicated server (VPS) environments. Existing cloud-level virtualization platforms are analyzed, and their limitations when deployed on small or specialized clusters are identified. A service-agent architecture and a text-based logic for determining the total workload of physical computing nodes are proposed. A software package has been implemented that is capable of automated resource scaling, launching backup server instances, and recovering system operations after hardware failures in an automated mode.

Keywords: VPS, virtualization, distributed systems, resource monitoring, automated scaling, gRPC, service-agent architecture.

Вступ

Сьогодні попит на хмарні обчислення та віртуальні виділені сервери (VPS) стрімко зростає. Коли IT-інфраструктура компанії розширюється, адмініструвати велику кількість фізичних серверів та сотні віртуальних машин вручну стає занадто складно. Це призводить до помилок через людський чинник, простоїв у роботі та неефективного використання потужностей комп'ютерів.

Великі промислові платформи (на кшталт OpenStack) розроблені для гігантських дата-центрів, тому вони надто громіздкі й складні в налаштуванні для малого та середнього бізнесу. Водночас простіші локальні системи керування віртуалізацією не мають вбудованих інтелектуальних функцій, які могли б самостійно реагувати на критичні зміни навантаження. Тому розробка

легковагової системи для автоматизованого керування, що самостійно стежить за серверами та приймає рішення про розподіл ресурсів, є дуже актуальною.

Архітектура системи та взаємодія компонентів

Створена система базується на сервісно-агентній моделі. Це означає, що вона розділена на дві основні частини:

1. Центральний контролер (керуючий рівень) – складається з окремих сервісів, які відповідають за авторизацію користувачів, створення й видалення VDS-серверів, збір статистики та збереження готових шаблонів операційних систем.

2. Агент вузла – спеціальна легка програма, яка встановлюється безпосередньо на кожен фізичний сервер, збирає дані про його роботу та виконує команди з центру.

Зв'язок між агентами та центральним контролером відбувається за допомогою сучасної технології gRPC. Агенти кожну секунду надсилають короткі повідомлення ("heartbeats"), підтверджуючи, що сервер працює, передають поточні метрики навантаження та миттєво забирають із черги нові команди на виконання. Для надійного збереження даних користувачів та журналів подій використовується база даних PostgreSQL, а для швидкого обміну оперативною інформацією в режимі реального часу – in-memory сховище Redis.

Механізм автоматичного керування ресурсами. Робота автоматики побудована на постійному аналізі трьох основних показників фізичних серверів: використання процесора (CPU), оперативної пам'яті (RAM) та мережевого трафіку. Система переводить ці показники у відсотки і розраховує загальний рівень завантаженості кожного хоста. При цьому найбільша увага (пріоритет) приділяється процесору та пам'яті.

Щоб система не смикалася через секундні випадкові стрибки навантаження, у логіку додано захисні затримки за часом. Тільки якщо сервер залишається сильно завантаженим протягом певного періоду, система запускає один із трьох сценаріїв захисту:

- Вертикальне масштабування: додає ядра процесора або об'єм оперативної пам'яті для конкретного VPS просто на поточному сервері, якщо там є вільне місце.
- Активація резерву: якщо на поточному сервері місця немає, система знаходить на іншому, менш завантаженому комп'ютері заздалегідь створений копію-сервер і вмикає його.
- Горизонтальне масштабування: автоматично створює і розгортає абсолютно новий віртуальний сервер із шаблону на найвільнішому хості кластера.

Окремим важливим сценарієм є реакція на аварії. Якщо від фізичного сервера перестають надходити сигнали життєдіяльності, система оголошує його недоступним і самостійно запускає процедуру аварійного відновлення – переносить конфігурації всіх віртуальних машин, що на ньому працювали, на інші живі сервери.

Основні елементи створеної системи, технології розробки та автоматичні сценарії детально систематизовано у таблиці 1.

Таблиця 1 – Порівняльний аналіз аналогів

Компонент / Проблема	Використані технології	Що виконує в автоматичному режимі
Центральне API та логіка	Python, фреймворк FastAPI	Керує доступом користувачів, обробляє запити від вебпанелі, віддає команди на сервери.
Зв'язок «Центр-Вузол»	Технологія gRPC, Protocol Buffers	Забезпечує миттєвий зв'язок, збирає щосекундні сигнали роботи серверів та дані про навантаження.
Перевантаження серверів	Модуль аналізу метрик (CPU/RAM)	Автоматично збільшує ресурси VPS, вмикає готові резервні копії або створює нові сервери з шаблону.
Аварійний збій хоста	Контролер тайм-аутів зв'язку	Виявляє раптове зникнення сервера з мережі та запускає аварійне перенесення віртуальних машин на живі вузли.
Оперативна пам'ять станів	In-memory база даних Redis	Миттєво оновлює та зберігає поточну карту всієї інфраструктури для виведення на екран адміна.
Головна база даних	СКБД PostgreSQL, SQLAlchemy	Надійно зберігає облікові записи, налаштування та детальний журнал усіх виконаних системою дій.
Панель адміністратора	Бібліотека React (JavaScript)	Візуалізує стан усієї мережі у вигляді зручних карток та графіків навантаження у реальному часі.

Для перевірки працездатності створений програмний комплекс було розгорнуто на тестовому стенді за допомогою Docker Compose. Тестування містило перевірку вебпанелі, контролера та трьох імітаційних серверів.

Під час штучного створення високого навантаження на одному з хостів система успішно зафіксувала проблему та автоматично додала віртуальній машині оперативну пам'ять і процесорну потужність, що стабілізувало її роботу. Під час імітації аварії (повного вимкнення одного з тестових серверів) система миттєво помітила відсутність зв'язку та самостійно перенесла роботу віртуальних машин на інші вільні сервери кластера, відобразивши успішний статус операції на графіках у панелі адміністратора.

Висновки

У роботі успішно вирішено завдання автоматизації адміністрування та підтримки стабільності розподілених VPS-систем. Створене програмне рішення дозволяє відмовитися від ручного моніторингу, самостійно виправляє проблеми з перевантаженням серверів та захищає інфраструктуру від втрати працездатності під час апаратних аварій. Це робить систему ефективним та доступним інструментом для побудови надійних приватних хмарних середовищ.

Список використаних джерел

1. Мухін В. Є. Хмарні технології : навч. посіб. Київ : Державний університет телекомунікацій, 2018. 150 с.

2. Жаріков Є. В. Хмарні обчислення : навч. посіб. Київ : КПІ ім. Ігоря Сікорського, 2019. 120 с.
3. OpenStack Open Source Cloud Computing Software. URL: https://www.openstack.org/.
4. Apache CloudStack Documentation. URL: https://cloudstack.apache.org/.
5. Proxmox Virtual Environment: Overview. URL: https://www.proxmox.com/en/products/proxmox-virtual-environment/overview.

AUTOMATION OF TRAFFIC FILTERING RULE GENERATION BASED ON THE RESULTS OF PASSIVE OSINT INFRASTRUCTURE ANALYSIS

Kaliakin Serhii Volodymyrovych
Senior Lecturer of the Department of Cybersecurity
Voropaiev Denys Vitaliiovych
Cadets of the Education and Research Institute No. 4
Kharkiv National University of Internal Affairs
Kamianets-Podilskyi, Ukraine

Modern cybercrime has evolved into the Cybercrime-as-a-Service (CaaS) model, which requires cybersecurity specialists to shift from passive incident logging to deep threat attribution. The primary challenge is the era of the "stealth adversary" and the high dispersion of digital footprints, which cannot be effectively analyzed using linear methods due to the high speed of attacker operations. According to recent analytical reports, a vast majority of current corporate infrastructure attacks are malware-free, occurring instead through the abuse of credentials, legitimate administration tools, and AI-driven reconnaissance automation [2]. To detect, model, and systematize such scattered and non-linear digital traces, it becomes essential to apply advanced graph modeling methods that visualize hidden connections between cyberspace entities and institutional infrastructure, relying upon comprehensive threat mitigation and cyber hygiene principles developed within specialized academic and institutional frameworks [3].

In light of this, this study focuses on developing an engineering solution for the direct conversion of passive OSINT reconnaissance results into dynamic configuration parameters of the Fail2ban defensive perimeter to ensure proactive network protection of an institution.

Within the scope of the research, the Maltego Community Edition (v. 4.10.1) toolset was utilized to analyze the infrastructure of the target domain. The process was based on the principles of Passive Reconnaissance, which implies data collection without direct interaction with the target's servers [4]. By applying sequential conversion operations (Transforms), specifically To Website [DNS], To DNS Name [Share], and To SSL Certificate, the internal topology of network assets was identified,