МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ІВАНОВ Ігор Валерійович

Математичне та програмне забезпечення автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу / Mathematical Tools and Software for Automatic Restoration of Information Services Based on Data From Network Monitoring Systems

спеціальність: 121 - Інженерія програмного забезпечення
освітньо-професійна програма - Інженерія програмного забезпечення

Кваліфікаційна робота

Виконав студент групи ІПЗм-21
І. В. Іванов

_____

Науковий керівник:
к.т.н., доцент Н. П. Порплиця

_____

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2022

ЗМІСТ

ВСТУП

Актуальність теми. Широке застосування обчислювальних інформаційних технологій зумовило напрямок науково-технічного прогресу та закріпило програмно-обчислювальне середовище, як невід'ємну частину всіх виробничих процесів та повсякденного життя сучасної людини. Об'єднання комп'ютерів в інформаційні мережі дозволило досягти високої продуктивності обчислень та переміщення на обчислювальну базу всього світового розмаїття традиційного виробничо-технологічного обладнання та приладів особистого користування. Сьогодні важко уявити, як людство могло обходитися без персонального комп'ютера чи смартфона, підключеного до глобальної мережі Інтернет.

Для організації ефективного управління інформаційною системою використовують моделі та методи аналізу станів обчислювальних ресурсів за отриманими даними від систем безперервного моніторингу. Під станом системи прийнято розуміти множину стійких значень змінних параметрів досліджуваного об'єкта. Дані моніторингу являють собою інформацію стану, яка є або первинною, або вторинною. Первинна інформація стану збирається безпосередньо з віддаленого об'єкта дослідження із заданою періодичністю вимірювання технологічних параметрів, тоді як вторинна інформація стану – це результати обробки даних на стороні сервера, зібраних з різних джерел, що прямо чи опосередковано стосуються досліджуваного об'єкта.

Зв'язок роботи з науковими програмами, планами, темами

Напрямок виконаних досліджень безпосередньо пов'язаний з науково-дослідним напрямком кафедри “комп’ютерних наук” Західноукраїнського національного університету.

Мета і задачі дослідження

Метою роботи є розробка методів та засобів автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

Відповідно до поставленої мети у роботі потрібно вирішити такі основні завдання дослідження:

1. Виконати аналіз сучасних підходів до організації хмарних обчислень, та розкрити суть організації спеціалізованих прогшрамних засобів на їх основі

2. Виконати оцінку продуктивності процесів управління комплексом за запропонованими ключовими показниками динамічного балансування хмарних сервісів

3. Розробити метод перенаправлення Інтернет-трафіку в резервну обчислювальну зону на основі розробленого алгоритму автоматичного перемикання.

4. Розробити інформаційну модель та метод моніторингу стану хмарних сервісів на прикладі Zabbix, який дозволили покращити управління обчислювальними потужностями у хмарній інфраструктурі.

5. Розробити програмне забезпечення для автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

6. Провести експериментальні дослідження розроблених методів та засобів.

Об’єкт дослідження – процеси відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

Предмет дослідження – методи та програмні засоби відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

Методи дослідження

В роботі використовувалися методи статистичного аналізу великих даних, теорії системного аналізу, методів об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів

Запропоновано метод динамічного розподілу навантаження на мікросервіс, який грунтується на метриці для вимірювання доступності та продуктивності сервісів і дозволяє динамічне оновлення за нульового часу простою.

Практичне значення одержаних результатів

В рамках виконання магістерського дослідження розроблено програмне забезпечення для управління сервісами на основі даних систем мережевого моніторингу типу Zabbix.

Особистий внесок магістранта

Всі результати отримані автором самостійно.

РОЗДІЛ 1
ОСОБЛИВОСТІ СУЧАСНИХ ТЕХНОЛОГІЙ ПОБУДОВИ ТА МОНІТОРИНГУ ХМАРНИХ ОБЧИСЛЮВАНИХ КОМПЛЕКСІВ

1.1. Різновиди інформаційних систем, побудованих у хмарі

Поняття хмарні обчислення визначає інформаційно-технологічну концепцію побудови системи, засновану на забезпеченні можливості повсюдного використання необхідних інформаційно-обчислювальних потужностей, через Інтернет або інші мережі передачі даних, що мають можливість швидкого розгортання або згортання з найменшими організаційно-експлуатаційними витратами. Серед таких ресурсів можуть використовуватися: пристрої накопичення даних, сервери, IT мережі передачі даних, програмне забезпечення або складно-ієрархічні програмні послуги.

Говорячи про різновид хмар (рисунок 1.1), необхідно встановити рівні віртуалізації ІС з погляду сервісів через Інтернет.

З множини форм прийнято виділяти 4 базових напрямів побудови ІС у компаніях, такі як:

• традиційна модель IT, у якій компанія відповідає за весь комплекс ІС самостійно –On Premises;

• обчислювальна інфраструктура як послуга – IaaS (Infrastructure as a Service);

• інформаційно-технологічна платформа як послуга – PaaS (Platform as a Service);

• програмне забезпечення як послуга – SaaS.

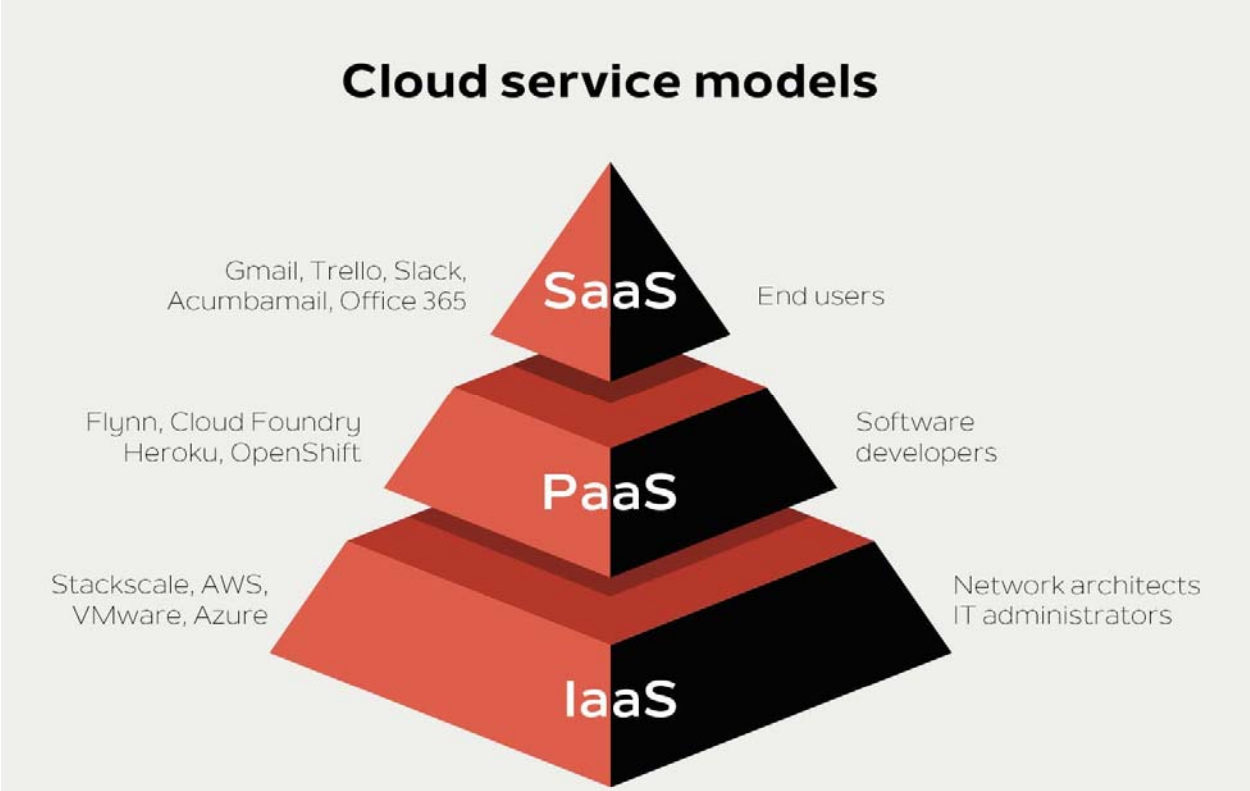

Рис. 1.1. Основні моделі хмарного сервісу

Перелічені напрями пов'язані між собою, але мають суттєві відмінності з погляду можливості контролю та використання комп'ютерних потужностей, що наочно продемонстровано рисунку 1.2.

У сучасному світі IaaS є базою, що постачаються через інтернет, скільки споживачам даної послуги надається широка свобода дій щодо використання обчислювальних потужностей, сховищ даних, IT мережевих пристроїв та спеціалізованого ПЗ, що відповідає за конфігурацію VM. У сучасному світі IaaS є базою, що постачаються через інтернет, оскільки споживачам даної послуги надається широка свобода дій щодо використання обчислювальних потужностей систем, сховищ даних, IT мережевих пристроїв та спеціалізованого ПЗ, що відповідає за конфігурацію VM.

Традиційний підхід	Сервісний підхід до управління IT інфраструктурою		
	IaaS	PaaS	SaaS
Установка патчів і оновлень протягом життєвого циклу програм	Установка патчів і оновлень протягом життєвого циклу програм	Установка патчів і оновлень протягом життєвого циклу програм	Установка патчів і оновлень протягом життєвого циклу програм
Завантаження Даних Користувача	Завантаження Даних Користувача	Завантаження Даних Користувача	Завантаження Даних Користувача
Установка складних додатків з багаторівневою архітектурою	Установка складних додатків з багаторівневою архітектурою	Установка складних додатків з багаторівневою архітектурою	Установка складних додатків з багаторівневою архітектурою
Установка патчів і оновлень протягом життєвого циклу Middleware and Runtime	Установка патчів і оновлень протягом життєвого циклу Middleware and Runtime	Установка патчів і оновлень протягом життєвого циклу Middleware and Runtime	Установка патчів і оновлень протягом життєвого циклу Middleware and Runtime
Установка додаткового ПЗ, бібліотеки, виконувані середовища: JAVA, .Net (Middleware and Runtime)	Установка додаткового ПЗ, бібліотеки, виконувані середовища: JAVA, .Net (Middleware and Runtime)	Установка додаткового ПЗ, бібліотеки, виконувані середовища: JAVA, .Net (Middleware and Runtime)	Установка додаткового ПЗ, бібліотеки, виконувані середовища: JAVA, .Net (Middleware and Runtime)
Установка патчів і оновлення в перебігу життєвого циклу ОС	Установка патчів і оновлення в перебігу життєвого циклу ОС	Установка патчів и оновлення в перебігу життєвого циклу ОС	Установка патчів и оновлення в перебігу життєвого циклу ОС
Установка і настройка ОС	Установка і настройка ОС	Установка і настройка ОС	Установка і настройка ОС
Установка гипервизора і настройка віртуалізації (опціонально)	Установка гипервизора і настройка віртуалізації (опціонально)	Установка гипервизора и настройка віртуалізації (опціонально)	Установка гипервизора и настройка віртуалізації (опціонально)
Виділення мережевих ресурсів (Фізичні порти, VLAN, IP адресації)	Виділення мережевих ресурсів (Фізичні порти, VLAN, IP адресації)	Виділення мережевих ресурсів (Фізичні порти, VLAN, IP адресації)	Виділення мережевих ресурсів (Фізичні порти, VLAN, IP адресації)
Виділення ресурсів Системи Зберігання Даних	Виділення ресурсів Системи Зберігання Даних	Виділення ресурсів Системи Зберігання Даних	Виділення ресурсів Системи Зберігання Даних
Виділення фізичного сервера	Виділення фізичного сервера	Виділення фізичного сервера	Виділення фізичного сервера

Рис. 1.2. Відмінності використання основних моделей хмарних сервісів

Таким чином, споживачі мають можливість контролювати ОС, встановлені на VM, накопичувачі даних, мережеві елементи, такі як програмні брандмауери FW (Firewall), пристрої розподілу та балансування

навантаження трафіку пакетів даних LBR (Load balancer) та інші; можливість встановлення будь-якого ПЗ та побудови закритих ІС. Іншими словами, IaaS надає об'ємний доступ до ІС постачальника через Інтернет для побудови інфраструктури за винятком можливості фізичного доступу до центру та контролю фізичного обладнання в ньому.

Послуга PaaS є інформаційно-технологічною платформою, побудованою в ІС провайдера, керовану провайдером. У ролі споживачів виступають розробники ПЗ, за певну плату та через інтерфейс прикладного програмування API вони отримують доступ до суто ієрархічного набору ПЗ та ІТ інфраструктур з обмеженою можливістю зміни компонентів ІС. Прикладами PaaS є такі послуги як: надсилання текстових повідомлень, використання засобів розробки та тестування, керування голосовими повідомленнями.

Серед великих провайдерів інформаційно-технологічних платформ вказуються такі компанії: Amazon.com, Salesforce.com, LongJump, Microsoft, IBM, Red Hat, VMWare, Google, CloudBees, Engine Yard.

Важливим напрямом стратегії розвитку ІТ послуг є ІС, побудовані за принципом SaaS, що дозволяють організаціям, незалежно від форми власності та розміру, використовувати програмне забезпечення для кінцевих користувачів за принципом аутсорсингу, коли для забезпечення внутрішніх потреб ІТ компанії залучаються зовнішні програмні продукти, побудовані за принципом хмарних обчислень.

Відмінною рисою SaaS є надання за певну плату програмного забезпечення для кінцевих користувачів через Інтернет без можливості контролю інфраструктури, прикладами програмного забезпечення є операційні системи VM, фізичне комп'ютерне обладнання та мережеві пристрої передачі даних.

1.2. Моделі та ризики архітектури побудови хмар

Для створення сучасних ІС, заснованих на хмарних обчисленнях (рисунок 1.3), використовуються 3 основні архітектурні моделі побудови хмар:

− Внутрішня хмара (Private/On Premises Cloud).

− Відкрита хмара (Public Cloud).

− Гібридна хмара (Hybrid Cloud).

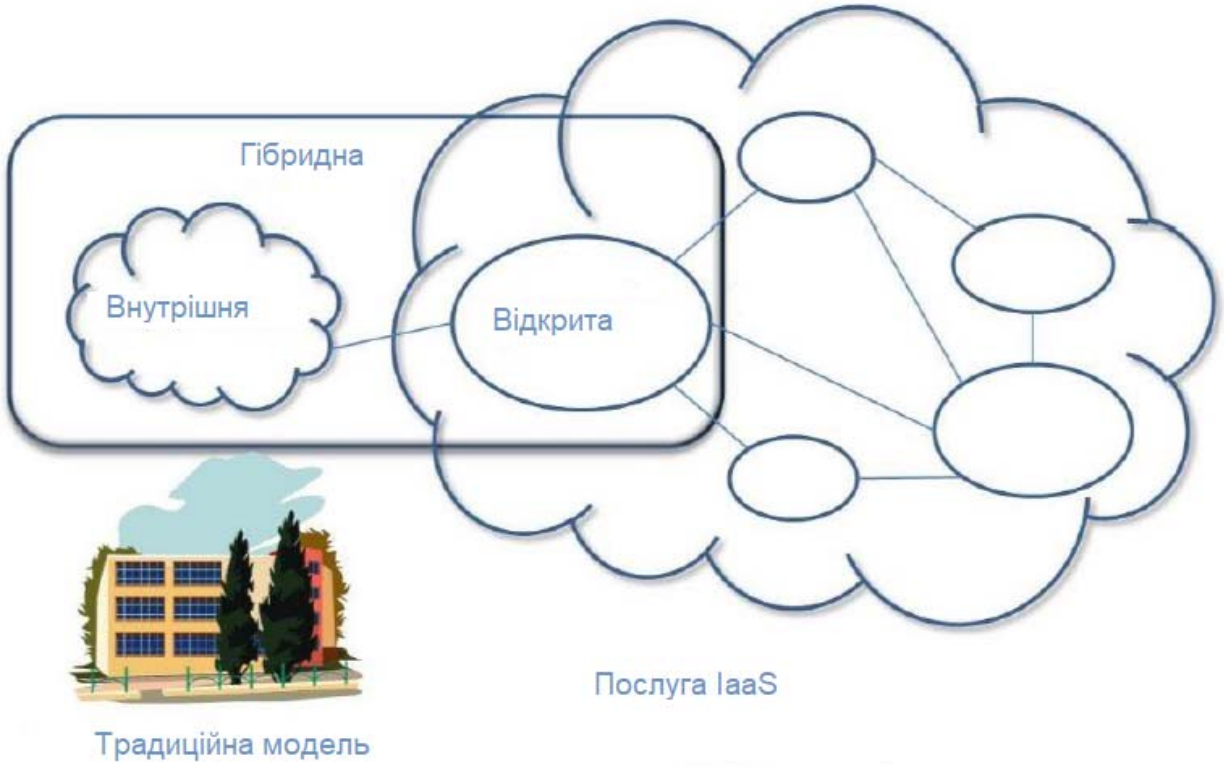

Рис. 1.3. Основні архітектурні моделі побудови хмар

Внутрішню хмару іноді називають власною чи корпоративною, тому що обладнання, на якому побудовано систему, встановлено на технологічних майданчиках, що належать компанії в центрах опрацювання даними, як власних, так і орендованих. Таким чином, компанія підтримує традиційну IT модель, що дозволяє мати повний контроль над комп'ютерно-обчислювальними ресурсами та іншими аспектами функціонування ІС,

такими як якість обслуговування кінцевих користувачів, розгортання та інсталяція нового ПЗ, супровід систем та аспектів IT безпеки тощо. Для технічної підтримки внутрішньої хмари компанія зобов'язана мати власну або зовнішню (аутсорсинг) IT службу, а також нести всі витрати, пов'язані з обслуговуванням центру.

З розвитком інформаційно-обчислювальних технологій компаніям для побудови корпоративних систем простіше скористатися послугами відкритої хмари, коли необхідні частини беруться в оренду у провайдера IaaS. Це виключає витрати обслуговування свого центру і висококваліфікований IT персонал, так як провайдер хмари у переважній кількості випадків надає весь спектр додаткових послуг, пов'язаних із резервним копіюванням даних, відновленням та підтримкою працездатності ІС у разі форс-мажорів та стихійних лих тощо. Такі хмари майже в 100% випадках використовують компанії-стартапи, де особливо важлива економія коштів, так як використання VM та інших ресурсів хмари може оплачуватись щохвилини.

Використання відкритої хмари має такі переваги:

• Скорочення IT-витрат на обслуговування внутрішніх локальних мереж;

• Зведення до мінімуму необхідного IT-персоналу для підтримки кінцевих користувачів та відсутність необхідності залучення висококваліфікованого IT-персоналу;

• Гнучка масштабованість ресурсів хмарних обчислень та планування необхідної обчислювальної потужності;

• Високо доступні обчислювальні потужності з резервним копіюванням, аварійним відновленням тощо.

Незважаючи на великі переваги технології відкритої хмари мають певні проблеми, кібербезпека є однією з них. Конфіденційна особиста інформація зберігається у зовнішній загальнодоступній мережі, яку підтримує постачальник, і передається по загальним каналам Інтернет. До проблем

також відноситься токенізація секретної інформації, шифрування даних під час передачі.

Гібридна хмара є інтеграційною ІС з перевагами і недоліками внутрішньої і відкритої хмар. В основному компанії використовують гібридну архітектуру з метою тимчасового підвищення ємності, коли додаткові обчислювальні ресурси можуть бути негайно розгорнуті у відкритій хмарі і додані у процеси клієнтських запитів, як у свою систему. Рідше подібна модель використовується для організації процесу резервного запису та зберігання даних для відновлення внутрішньої хмари у разі настання непередбачених обставин та втрати функціональності власної інфраструктури.

У таблиці 1.1 наводиться порівняльний аналіз використання моделей архітектури системи щодо послуг, які надаються на основі хмарних обчислень.

Таблиця 1.1.

Практичне застосування моделей інформаційних систем та послуг на основі хмарних обчислень

Види хмар	Внутрішні	Гібридні	Відкриті
SaaS	Компанія здійснює внутрішню розробку ПЗ та надає його кінцевому користувачеві	Компанія використовує як внутрішнє, так і зовнішнє ПЗ, що надається через Інтернет	Компанія використовує готове прикладне програмне забезпечення через інтернет
PaaS	Компанія обслуговує власну систему для розробки програмного забезпечення третіми компаніями	Компанія управляє різнорідними системами розробки як власного ПЗ, і третіми компаніями	Вендор надає систему для розробки ПЗ
IaaS	Компанія повністю обслуговує корпоративну систему	Вендор надає компанії ІС одноосібне користування для розширення власного комплексу	Компанія будує хмару на відрито використовуваній ІС

Підприємства IT галузі зацікавлені у скороченні витрат та збільшенні продуктивності, що дозволяє підвищити прибуток. Тому з появою хмарних технологій та послуг відкритих хмар багато учасників ринку виявляють інтерес щодо впровадження у себе на підприємствах нових технологій.

Зважаючи на відсутність відповідного досвіду та необхідних знань, існуючого IT персоналу, підприємства стикаються з низкою проблем, які заважають їм досягти бажаного результату в галузі економії витрат та спрощення процесів обслуговування ІС.

Технології відкритих хмар з'явилися порівняно недавно, для використання їх на підприємствах необхідно вивчити всі нюанси, пов'язані з безпекою та витоком цінної інформації, необхідно досліджувати раціональність їхнього застосування. У порівняльній таблиці 1.2 представлена узагальнена інформація, що показує переваги та недоліки двох видів хмаро, внутрішніх та відкритих виходячи з основних функцій.

Таблиця 1.2.

Переваги та недоліки між внутрішніми та відкритими хмарними системами

Функція	Внутрішні	Відкриті
Кваліфікація персоналу	Великі витрати на висококваліфікованих фахівців	Фахівці з необхідними знаннями та досвідом
Підтримка	Цілодобова робота обслуговуючого персоналу, витрати на кастомізацію систем моніторингу під власні потреби	Коробкові рішення для моніторингу, мінімальна кількість обслуговуючого персоналу для підтримки рівня додатків
Маштабування	Постійне дообладнання та придбання дорогих програмних продуктів	Можливість розширення/зменшення ІС відповідно до попиту на ресурси
Безпека	Додаткові витрати на придбання систем безпеки та проведення аудитів	Безпека забезпечується постачальниками послуг за пропорційну плату

Слід зазначити, що з використанням технологій відкритих хмар скорочуються витрати підприємства через відсутність необхідності ведення непрофільної діяльності щодо встановлення серверного та комунікаційного обладнання, витрат на його обслуговування із залученням додаткових висококваліфікованого ІТ персоналу. У разі купівлі обладнання та побудови власної ІТ інфраструктури компанія витрачає час на пошук постачальника, замовлення обладнання та доставку, доставка може займати кілька тижнів. Крім цього, потрібен час на встановлення та налаштування обладнання, сервісів та їх тестування.

Аналізуючи отримані дані світових дослідницьких агентств, можна дійти висновку у тому, що хмарні системи піддаються ризикам. Простота нарощування кількості VM призводить до ефекту експоненційного збільшення обсягу даних як службових, так і користувацьких, що може призвести до неможливості якісного обслуговування ІС. Методи управління ризиками показують, що ключовим рішенням підтримки працездатності ІС є надійна система безперервного моніторингу, здатна обслуговувати великі дані. Отже, перш ніж розпочинати впровадження таких технологій, необхідно провести моделювання нової системи та оцінити витрати на її обслуговування, а також виконати проектування відповідної системи безперервного моніторингу.

Перевага хмарних технологій полягає у коротких термінах запуску проектів, що змушує прогресивні ІТ компанії впроваджувати відповідні інструменти управління ризиками. Це дозволяє отримувати переваги у вигляді надійності та безпеки. Постачальник таких сервісів гарантує мінімізацію простоїв сервісу за рахунок використання резервного майданчика з обчислювальними потужностями. На відміну від варіанта купівлі обладнання, компанія може розширювати ряд необхідних обчислювальних ресурсів.

Для вітчизняних підприємств хмарні технології є відносно новим видом організації, що викликає недовіру з боку фахівців ІТ. Компанії, які

вирішили впровадити такі технології, найчастіше намагаються автоматизувати відразу всі послуги і не домагаються своєї мети щодо повернення витрачених інвестицій або отримання прибутку.

1.3. Огляд засобів моніторингу та управління інформаційними системами

Для організації ефективного управління супутніх робочих процесів з підтримки працездатності системи і планування ресурсів підприємства, що використовуються, IT компанії потребують засобів, в тому числі, що дозволяють автоматизувати планово-попереджувальний ремонт, побудову нових або додаткових VM, а також спростити інсталяцію нового ПЗ на всі технологічні рівні ІС.

Таким чином, прийнято розділяти спеціалізовані програмні засоби за двома основними категоріями: бізнес-додатки, орієнтовані на стратегію планування всіх видів ресурсів ERP (Enterprise Resource Planning) та технологічні засоби управління системами, що використовуються IT підрозділами, що відповідають за такі функції, як:

• Інсталяція нового програмного забезпечення в рамках SDLC (Software Development Life Cycle), включаючи критичні оновлення, пов'язані, зокрема, з аспектами безпеки;

• Контроль та безперервний моніторинг продуктивності комплексу та планування його ємності;

• Обслуговування VM, побудованих на різних різновидах операційних систем та програмного забезпечення, включаючи їх своєчасне оновлення на нові версії;

• Резервне копіювання даних, VM та їх відновлення у разі виникнення позаштатних ситуацій з працездатністю, а також його повною чи частковою втратою;

• Управління доступом клієнтів до IT сервісів та ресурсів ІС.

Практика компаній, що управляють такими комплексами, показує, що використання IT-фахівцями автоматизованих засобів моніторингу та управління допомагає суттєво зменшити кількість рутинних трудомістких ручних операцій, що дозволяє стандартизувати операційну діяльність, мінімізувати кількість інцидентів, пов'язаних з людським фактором, а також підвищити надійність надання IT- послуг.

Порівняльний аналіз систем моніторингу глобально розподілених обчислювальних комплексів показав, що архітектура різних засобів управління суттєво не відрізняється один від одного та містить серверну частину, що підтримує консоль управління для IT фахівців, які обслуговують ІС. На керовані компоненти встановлюється ПЗ, що представляє собою програмні додатки або служби (так звані агенти), які здатні виконувати команди, що визначаються центральною серверною частиною. Агенти додатково збирають параметричні відомості про кожну компоненту і передають їх на центральний процесор УПС, де відбувається обробка отриманої інформації та подальше відображення подій, що відбуваються на графічному інтерфейсі консолі управління.

На основі зібраних даних автоматизовані засоби управління самостійно здійснюють різноманітні операції [1,2,3-6] без участі обслуговуючого персоналу. Прикладом таких автономних дій може бути запуск діагностичного тестування VM; відновлення працездатності окремої VM, бізнес-додатку чи цілого сегменту; розгортання додаткових VM для підтримки ємності ІС під час години найбільшого навантаження та багато іншого.

1.4. Класифікація систем моніторингу

В основу класифікації систем моніторингу покладено різні знання, до яких належать: цілі проведення моніторингу; його основні функції; сферу

застосування даних; інструментарій; модель чи технологія проведення моніторингу та ін.

Найчастіше системи моніторингу ІС класифікують відповідно до їх основних функцій: інформаційної, діагностичної, порівняльної та прогностичної. Щоб визначити який вид моніторингу потрібен, важливо визначити, який процес необхідно відстежити. І вже від цього вибирати вид систем безперервного моніторингу.

Розглянемо три основні види систем моніторингу:

1) Моніторинг серверів. Дозволяє одночасно спостерігати за всіма серверами у комплексі. Оперативно створює оповіщення ще до виникнення проблем. Дозволяє скоротити час простою серверів, контролювати цілодобову роботу серверів та мережі дозволяє значно підвищити їх продуктивність. Постійне відстеження параметрів VM забезпечує швидке виявлення та усунення несправності – до того, як її помітить користувач. Системи моніторингу цього типу надають інформацію про продуктивність, показують необхідність розширення ємності системи. База даних результатів опитування, зібрана за час роботи служби моніторингу серверів, дає можливість проаналізувати та виявити потенційні проблеми у каналах зв'язку та устаткуванні завчасно. Низка проблем може бути усунена системою моніторингу автоматичним шляхом перезавантаження певних служб та серверів без залучення мережевих адміністраторів. Якщо поломка об'ємна та потребує участі технічного персоналу, то зібрана в динаміці інформація значно полегшує роботу та зменшує час усунення несправності у 2-3 рази. Несуттєва поломка або перебій у наданні сервісів, з погляду клієнта ІС, може спричинити суттєві фінансові втрати, невдоволення та втрату користувачів. Тому, незважаючи на трудомісткість установки системи моніторингу серверів, будь-якому великому центру опрацювання даних вона необхідна.

За графіками завантаження системи, які формує служба моніторингу серверів, можна скласти критерії оптимальності витрати обчислювальних потужностей. При хмарних обчисленнях у моменти мінімального

завантаження є можливість відключати деякі сервери або переводити їх у режим очікування, це дозволяє суттєво скоротити витрати на утримання дата центру. А за динамікою зростання навантаження легко заздалегідь спроектувати обов'язкове збільшення потужності хмарного дата центру, яке необхідне задоволення потреб клієнтів через кілька місяців.

2) Моніторинг мікросервісів необхідний для забезпечення більшої функціональності та надійності створення програмного забезпечення. Ця зміна призвела до хвильового ефекту в рамках створення спеціального програмного забезпечення для управління системою, включаючи системи моніторингу. Безперервний моніторинг показників мікросервісів може включати в себе запит на секунду, доступну пам'ять, невдалу автентифікацію, потоки, з'єднання тощо. Можна використовувати систему такого моніторингу у відкритих хмарах, щоб контролювати швидкість передачі. Моніторинг мікросервісів дає змогу це реалізувати, що дозволяє обмежити кількість підключень, запитів, які можуть зробити користувачі ІТ сервісів. Наприклад, це кількість підключень, які можна зробити клієнту за певний період часу, після того як він пройде систему ідентифікації. Це свого роду захист від навантаження обчислювальних ресурсів комплексу.

Моніторинг є важливою частиною мікросервісу так, чим складніше ПЗ, тим складніше зрозуміти його роботу і усунути недоліки. Мікросервіс – це логічне продовження поділу програм та коригування ресурсів програми для завантаження за допомогою горизонтального масштабування. Це підхід до розробки єдиної програми як набору невеликих послуг, кожна з яких працює у своєму власному процесі та спілкується з деякими механізмами, часто за допомогою HTTP ресурсів. Контейнери – це будівельні блоки мікросервісу. Коли у них виникають помилки необхідно детально вивчити, що відбувається всередині них, у тому коді, тобто потрібна глибока внутрішня діагностика контейнерів. Цим займається моніторинг мікросервісу. Також це стосується і інформації на рівні додатків, наприклад, у тому, які запити мають найбільший час відгуку.

На рисунку 1.4 наведено приклад моніторингу дев'яти мікрослужб, що взаємодіють один з одним, які в свою чергу надають чотири інтерфейси для різних додатків, для інтерфейсу користувача і для апаратної частини хмари. Мікросервіс буде те, що виконує тільки одну функцію, через API (Application Programming Interface).

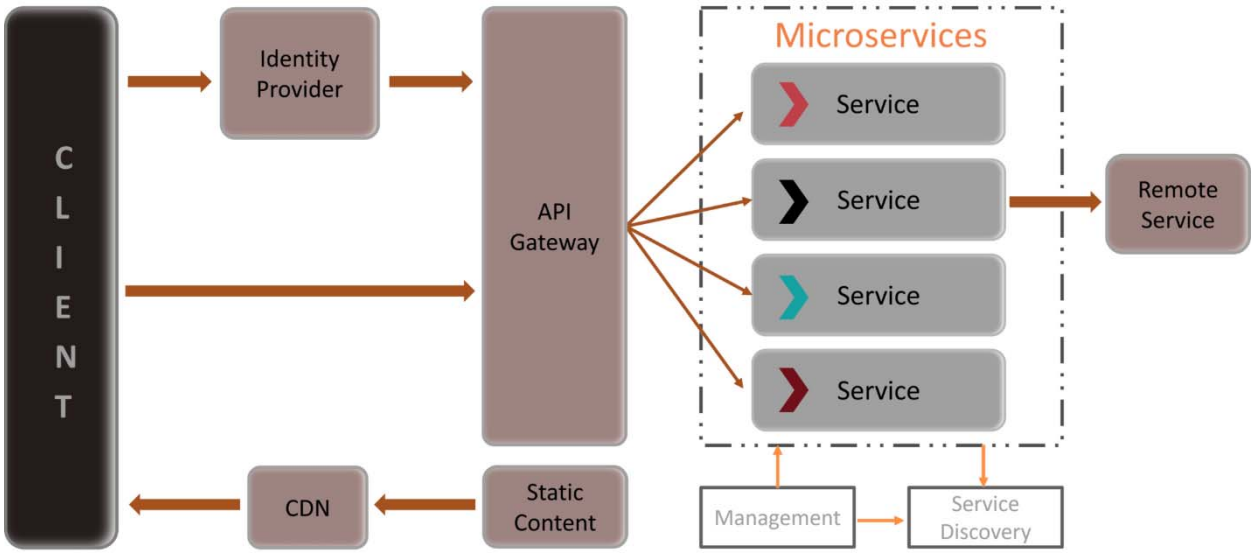

Рис. 1.4. Схема побудови моніторингу мікросервісів

Також використовують метрики журналу для моніторингу своїх мікросервісів. За допомогою такого моніторингу можливо, наприклад, переглянути час відгуку та побачити окремі транзакції обчислень. Також моніторинг дозволяє розібратися з тим, що відбувається з програмами з точки зору апаратної частини хмари.

3) Моніторинг стану всієї IT інфраструктури ІС. Такий моніторинг дозволяє IT системним адміністраторам отримувати сповіщення про несправності до того, як вони вплинуть на кінцевого споживача. Програмний комплекс моніторингу IT інфраструктури призначений для контролю працездатності хмари, включаючи мережеве та серверне обладнання, дискові масиви, джерела безперебійного живлення, робочі станції, обладнання офісної IP-телефонії. Система моніторингу дозволяє швидко та ефективно

реагувати на появу збоїв у роботі мереж зв'язку, серверів, обладнання та запобігати їх виникненню. Відмінна риса даної системи – спеціалізація на контролі апаратного забезпечення, робота за стандартними інтерфейсами та протоколами.

Інструменти моніторингу IT інфраструктури дозволяють організаціям виявляти та вирішувати проблеми, перш ніж вони можуть негативно вплинути на критично важливі бізнес-процеси. Вони дають уявлення про стан фізичних, віртуальних та хмарних систем, забезпечують їх доступність та продуктивність. Крім моніторингу всієї інфраструктури та бізнес-процесів, програмні засоби також можуть допомогти у плануванні модернізації апаратної частини центру, перш ніж застарілі системи почнуть викликати збої та проблеми життєздатності до того, як ці проблеми стануть актуальними.

За допомогою засобів моніторингу інфраструктури можна побачити, як програмні процеси взаємодіють між собою, включаючи споживання інтерфейсних ресурсів, а також аномалії та збої в процесах. За допомогою такого моніторингу можна відстежити продуктивність кожного шару системи та зрозуміти, наскільки добре сервери виконують запити, особливо під постійним високим навантаженням. Як і в інших видах моніторингу, в цьому виді збираються метрики, які можна візуалізувати, перетворити в часові ряди, трендові значення та інші дані.

Всі системи управління та моніторингу ІС служать, щоб знизити роль людського фактора при експлуатації IT інфраструктури. Система моніторингу має давати чітку відповідь на питання, пов'язані з ефективністю функціонування підсистеми IT інфраструктури. Поряд із зниженням наявних витрат IT компаній на інформаційні технології, системи управління та моніторингу ІС забезпечують подальший розвиток IT інфраструктури у виваженому та стратегічно правильному напрямку. Керівництво компанії має розуміти, які процеси відбуваються на сучасному рівні розвитку IT інфраструктури та яка модернізація необхідна у перспективі.

1.5. Аналіз систем моніторингу з відкритим програмним кодом для хмарних інформаційних технологій

Сучасні системи безперервного моніторингу засновані на постійному зборі та аналізі даних про стан серверів, що описуються заданим набором системних метрик (використання ресурсів процесора, оперативної та зовнішньої пам'яті тощо). Автоматизація моніторингу дозволяє скоротити витрати на адміністрування хмарної інфраструктури, унеможливити вплив людського фактора, знизити час обслуговування системи.

У таблиці 1.3 наведено порівняльні характеристики систем моніторингу серверів.

Таблиця 1.3.

Порівняльні характеристики систем моніторингу серверів

Характеристика	Nagios	Zabbix	Graphite	New Relic	Splunk
Кількість серверів/метрик без затримок	400/103	105/106	103/105	104/105	10 ТВ/день
Ліцензія	безкоштовно	безкоштовно	умовно безкоштовно	платна	безкоштовно/ 500 Mb на день
Налаштування конфігурації моніторингу	середньо: необхідна адаптація	легко: вбудовані шаблони	складно: кодування надбудови	легко: вбудовані шаблони	легко: через веб-інтерфейс
Мови програмування налаштувань	C, Perl, Python, Shell scripts	C# і Python	Python, C, PHP, Ruby, .Net, Java	Ruby, Java, PHP, Python, .Net,	Python, C#, Java, JavaScript, PHP, Ruby
Типи СУБД, що підтримуються для зберігання даних моніторингу	MySQL	MySQL, PostgreSQL, Oracle, IBM DB2, NoSQL	Whisper	Sharded MySQL, Percona build	Файли
Контрольовані операційні системи віддалених серверів	Windows, Linux, FreeBSD	Windows, Linux, MacOS, FreeBSD, IBM AIX	Windows, Linux, MacOS	Windows, Linux	Windows, Linux, MacOS
Прогнозування	Так	Так	Так	Так	Так
Автоматичне відновлення	Так	Так	Ні	Так	Так

Nagios (рисунок 1.5) – найпоширеніша система моніторингу, що має спільноту чисельністю 1 млн. користувачів для обміну інформацією [7-9]. Nagios Core – це безкоштовний продукт із відкритим кодом, його можна адаптувати під потреби компанії, вбудовувати додаткові модулі та керувати ними через веб-інтерфейс. Важливою особливістю продукту є можливість прогнозування найпростіших відмов та їх автоматичного відновлення без участі людини.

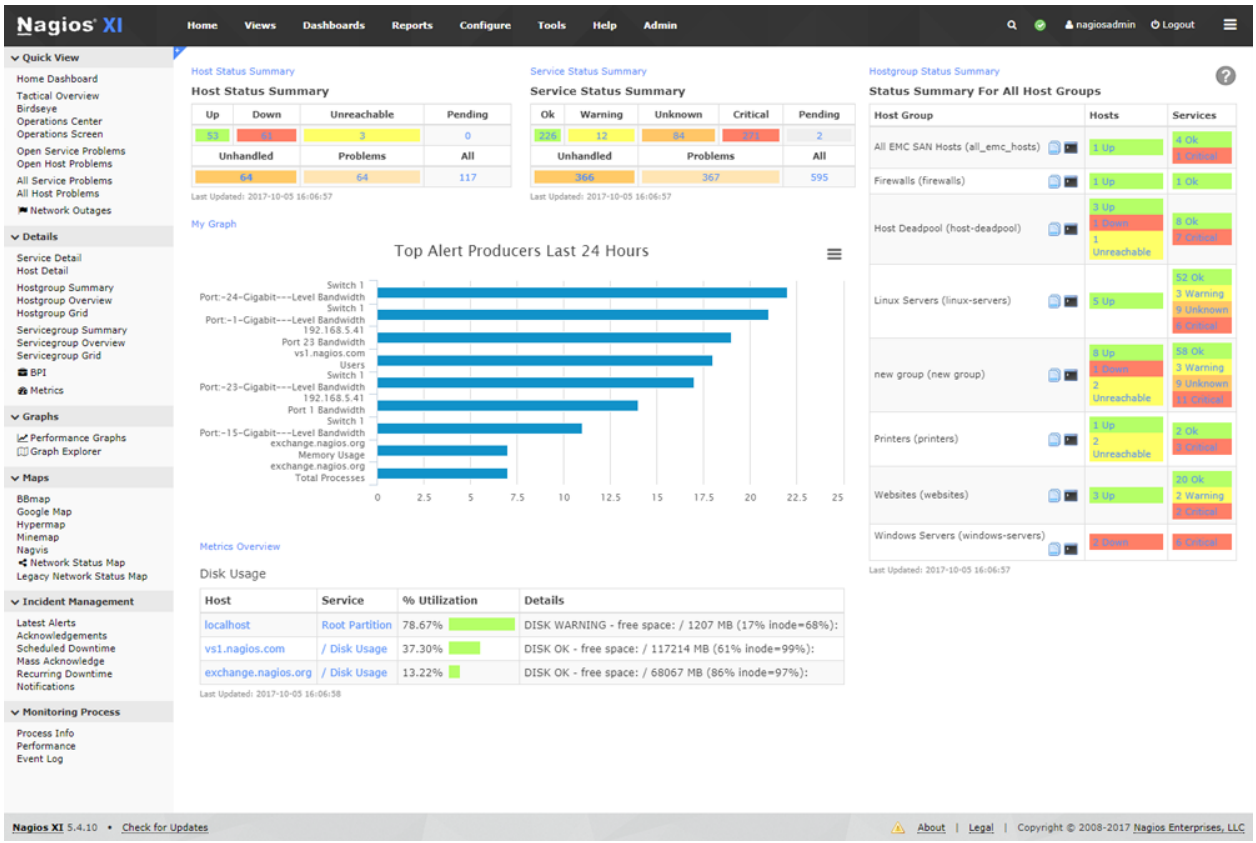

Рис. 1.5. Система Nagios

Головна проблема Nagios полягає у масштабованості. Всі дані з серверів збираються в одну чергу, і зі збільшенням їх кількості (з досягнення 400 серверів по 5 перевірок на кожен) відбувається запізнення обробки та візуалізації даних.

Zabbix (рисунок 1.6) – продуктивна система моніторингу масштабу підприємства з відкритим кодом та широким функціоналом, що успішно

конкурує з платними рішеннями. Продукт спеціалізований на системному моніторингу, підтримує всі операційні системи (ОС), дозволяє оперативно включити сервери до моніторингу за вбудованими шаблонами. Для зберігання історичних даних є широкий вибір реляційних систем управління базами даних (СУБД) та NoSQL, можлива архітектурна оптимізація [9]. В останніх версіях Zabbix передбачено функції прогнозування [10-14]. Проблема спаму вирішується в Zabbix шляхом налаштування порогів спрацьовування тригерів, встановлення між ними залежності, встановлення часу профілактики серверів.

Рис. 1.6. Система Zabbix

Graphite (рисунок 1.7) - система моніторингу з відкритим вихідним кодом. Продукт використовує власну БД Whisper [15-20] та складний у налаштуванні [21], що є недоліком. Однак це компенсується високою продуктивністю при обробці великих даних (105 VPS із 103 серверів у реальному часі). Система розроблена компанією Orbitz для власного моніторингу системних та бізнес-метрик, реалізованих за допомогою надбудов, у тому числі прогностичних.

Рис. 1.7. Система Graphite

Недоліками Graphite є специфіка реалізації функцій під конкретні бізнес-процеси та відсутність автоматичного відновлення після виявлення відмови.

New Relic (рисунок 1.8) відома як система моніторингу додатків, але надає рішення і для системних індикаторів із вбудованою підтримкою фільтрації хибних сповіщень та прогнозування відмов. За даними розробника [22], близько 20 млн користувачів використовують продукти New Relic, які доступні у відомих хмарних середовищах Amazon та Azure.

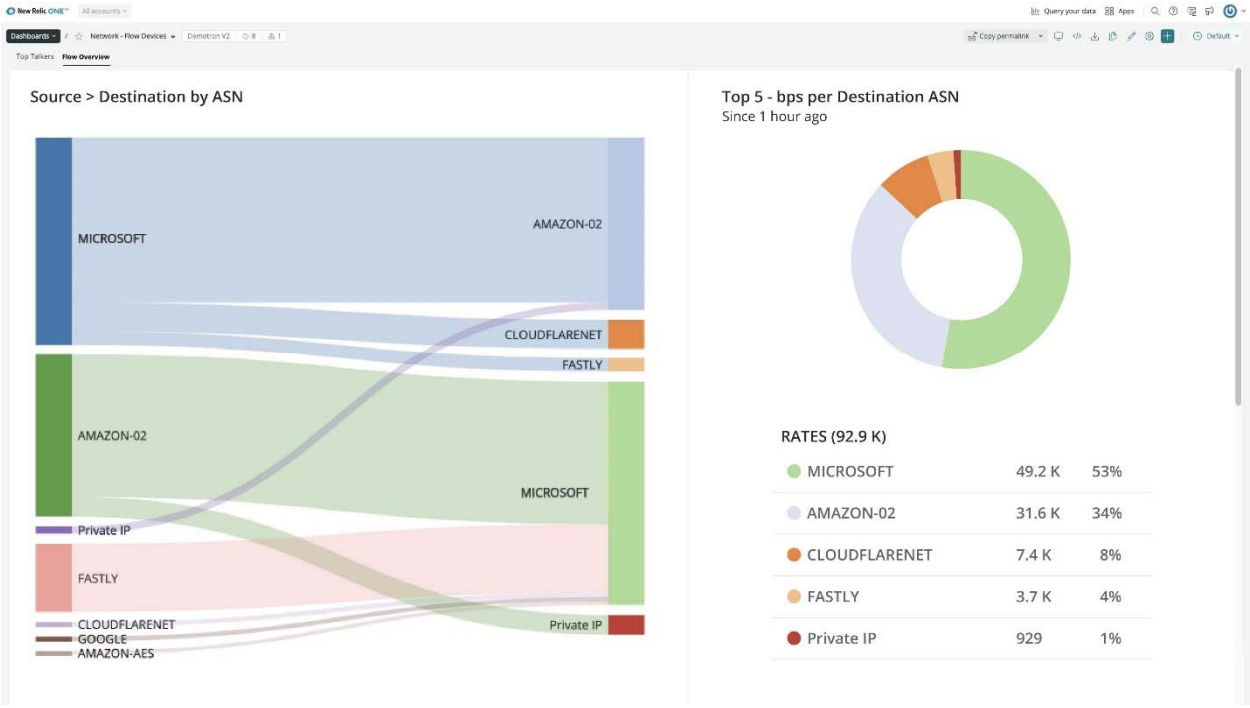

Рис. 1.8. Система New Relic

З іншого боку, вимоги безпеки компаній не дозволяють зберігати дані, у тому числі у зашифрованому вигляді, на зовнішніх серверах у хмарному середовищі. З урахуванням цього недоліку New Relic надав можливість підключати свої модулі методом JSON HTTP POST та отримувати дані із серверів.

Splunk (рисунок 1.9) пропонує кілька продуктів, платних, з великими можливостями моніторингу та аналізу зібраних даних. Splunk Cloud – це рішення для моніторингу хмарної інфраструктури, доступне у сервісах Amazon [24-30]. Існують готові програми та можна розробляти власні, які збирають дані в систему Splunk, де реалізовані ефективні засоби аналізу, візуалізації, прогнозування, запобігання відмовам та інші.

У зв'язку зі швидким зростанням та глобалізацією хмарних сервісів при виборі системи моніторингу на перше місце виходять такі критерії, як прогнозування подій на основі історичних трендів, автоматичне відновлення сервісів після відмови, фільтрація хибних сповіщень, особливо під час планової профілактики. Надмірний обсяг спам-повідомлень викликає шум на графічному інтерфейсі чергової зміни GNOC, на тлі якого можна пропустити

справжню відмову. Реалізація цих функцій є пріоритетним напрямом підвищення ефективності моніторингу.

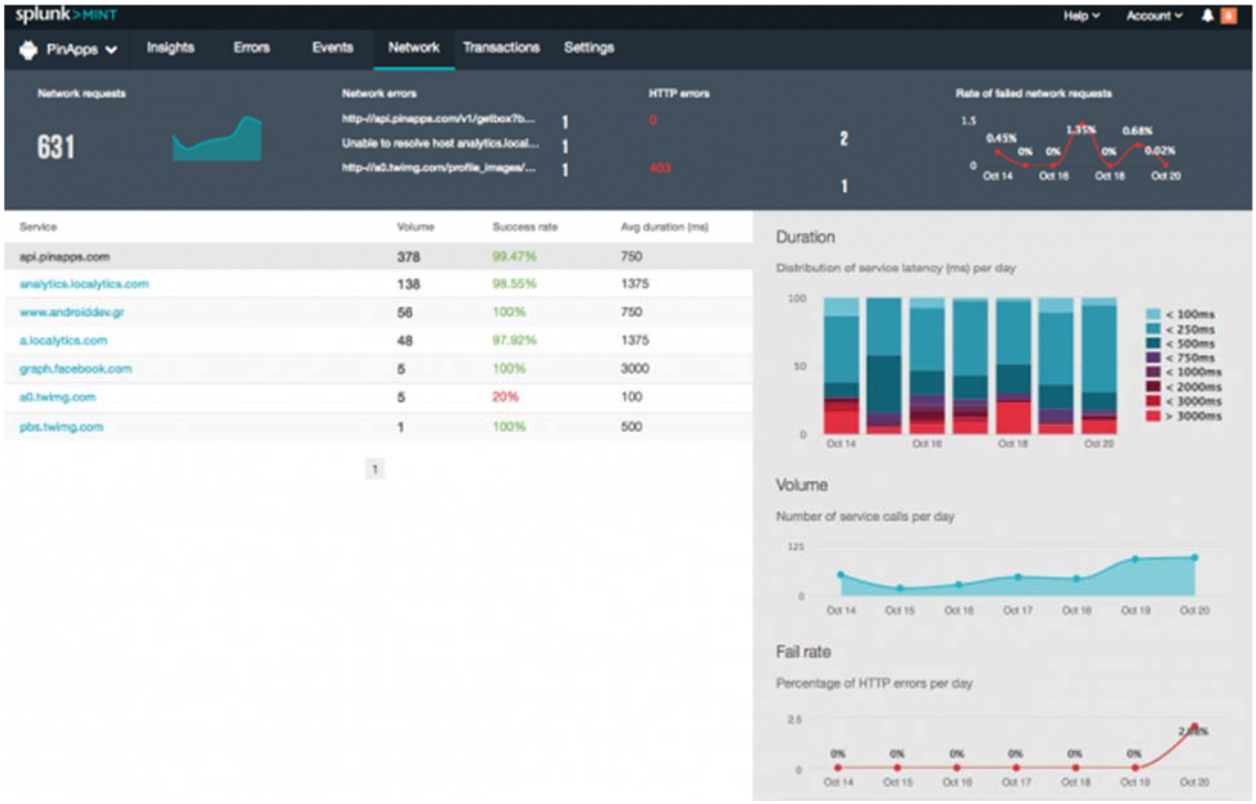

Рис. 1.9. Система Splunk

1.6. Постановка задачі дослідження

Метою роботи є розробка методів та засобів автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

Відповідно до поставленої мети у роботі потрібно вирішити такі основні завдання дослідження:

1. Виконати аналіз сучасних підходів до організації хмарних обчислень, та розкрити суть організації спеціалізованих прогшрамних засобів на їх основі

2. Виконати оцінку продуктивності процесів управління комплексом за запропонованими ключовими показниками динамічного балансування хмарних сервісів

3. Розробити метод перенаправлення Інтернет-трафіку в резервну обчислювальну зону на основі розробленого алгоритму автоматичного перемикання.

4. Розробити інформаційну модель та метод моніторингу стану хмарних сервісів на прикладі Zabbix, який дозволили покращити управління обчислювальними потужностями у хмарній інфраструктурі.

5. Розробити програмне забезпечення для автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

6. Провести експериментальні дослідження розроблених методів та засобів.

Висновки до першого розділу

1. Проаналізовано переваги і недоліки організації систем з використанням хмарних технологій.

2. Розглянуто особливості засобів моніторингу систем управління на основі хмарних сервісів.

3. Виконано аналіз існуючих систем моніторингу з відкритим вихідним програмним кодом для хмарних інформаційних технологій.

РОЗДІЛ 2
МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЧНОГО ВІДНОВЛЕННЯ ІНФОРМАЦІЙНИХ СЕРВІСІВ НА ОСНОВІ ДАНИХ СИСТЕМ МЕРЕЖЕВОГО МОНІТОРИНГУ

2.1. Процес виявлення аномалій у працездатності інформаційних послуг

Раніше Інтернет провайдери забезпечували користувачам канали зв'язку та передачу потоків даних. На даний момент намітилася тенденція, коли телекомунікаційні компанії надають доступ до внутрішніх ресурсів IaaS та/або інформаційних сервісів SaaS, що реалізуються за хмарними технологіями віртуалізації.

У досліджуваних архітектурах функціонування всієї глобальної хмарної інфраструктури відстежується спеціальним оперативним центром GNOC, який працює в режимі 24/7 та несе відповідальність за безперебійність роботи хмарних сервісів відповідно до ухваленої угоди про рівень обслуговування SLA понад 300 тис. корпоративних клієнтів. Дані та критичні події на численних віддалених серверах відстежуються моніторинговою системою Zabbix в режимі реального часу, в єдиній БД і автоматично відображаються веб-додатком на централізованому табло NMC з метою швидкого виявлення та реагування на можливі аномалії.

Аналіз оперативних та історичних даних в СУБД Zabbix дозволяє оцінити динаміку, визначити аномалії, що повторюються, передбачити і заздалегідь запобігти багатьом проблемам, не допускаючи повної відмови розподілених сервісів в обслуговуванні DDoS (Distributed Denial of Service).

У таблиці 2.1 наведено джерела інформації та методи відстеження критичних подій у досліджуваних системах.

Таблиця 2.1

Методи виявлення аномалій у компанії

Джерела та способи виявлення аномалій	Середня статистика
Автоматичні оповіщення в моніторинговій системі Zabbix	90%
Повідомлення від зовнішніх партнерів та постачальників Інтернет-послуг	4%
Функціональні тести, що виконуються вручну із заданою періодичністю	3%
Повідомлення від служби підтримки корпоративних клієнтів	2%
Інші джерела	1%

У разі виявлення аномалії, дії GNOC включають такі основні етапи:

1) виявлення та підтвердження аномалії за допомогою NMC або одним із способів, зазначених у таблиці 2.1;

2) повідомлення інших членів команди обслуговуючого технічного персоналу та протоколювання повідомлень про помилку в одній із систем документування.

3) відновлення системи наявними автоматизованими засобами GNOC, якщо проблема добре відома і для неї передбачено оперативне рішення, наприклад, перезавантаження VM або перенаправлення робочого навантаження на запасний мережевий або обчислювальний ресурс;

4) коли функціонал системи не може бути відновлений засобами GNOC, інцидент ескалується інженерам служби технічної підтримки. Якщо у процесі вирішення проблеми модифікується програмне забезпечення або інфраструктура, то зміни відображаються в окремій інформаційній системі власної розробки CMP (Change Management Portal), пов'язаної із системою планування проектів Roadmap.

Для ряду хронічних проблем, що належать до зовнішніх постачальників і не залежать від внутрішніх ресурсів IT компаній, передбачено автоматичні процедури відновлення працездатності. У таких випадках участь GNOC на етапах 1 і 2 не потрібна.

Прикладом такої реалізації є автоматичне відновлення Java сервісів в результаті загальновідомої проблеми витоку пам'яті при піковому навантаженні на додатки з метою запобігання DDoS.

2.2. Ключові показники ефективності процесів керування хмарними сервісами

Оцінюючи ефективність процесів в Zabbix/Roadmap за допомогою KPIs, головна проблема полягає в розрізненості перелічених джерел даних, що зберігаються в окремих СУБД. Оскільки в СУБД моніторингової системи Zabbix сконцентрована вся інформація про віддалені сервери, було запропоновано новий метод використання вбудованих засобів Zabbix для інтеграції СУБД, що використовуються, і автоматизації обчислень KPIs. З цією метою розроблені гетерогенні SQL запити, які надсилаються за розкладом з проксі-сервера Zabbix на розподілені сервери БД інших IT сервісів, результат їх виконання повертається в Zabbix як окремі значення метрик по кожній із систем.

У таблиці 2.2 як приклад наведено деякі метрики та їх щоденні значення для системи IMP. Інтеграція даних у Zabbix згодом дозволяє створювати нові метрики на основі агрегованих виразів будь-якого ступеня складності, які вже обчислюються локально на стороні сервера Zabbix і зберігаються в його СУБД.

Для інтегральної оцінки QoS у досліджуваних архітектурах реалізовані такі метрики KPIs, показані на схемі рисунку 2.1:

1) час виявлення аномалії. Традиційно визначається як різниця між фактичною подією на віддаленому сервері та часом його фіксації у моніторинговій системі. Насправді існує незначна затримка, викликана об'єктивною потребою передачі останніх даних з віддаленого сервера через проксі-сервер у СУБД моніторингової системи Zabbix. Тому в IT компаніях

прийнято оцінювати як час реакції GNOC на критичну подію NMC з максимальним SLA 2 хвилини;

Таблиця 2.2

Приклад щоденного звіту щодо інцидентів у Zabbix

Найменування метрики у Zabbix	Значення за останній день
% інцидентів, автоматично виявлених у Zabbix	84
% інцидентів, ескалованих до служби підтримки	0
% інцидентів за результатами ручних годинних тестів	11
% інцидентів, виявлених корпоративними клієнтами	0
% інцидентів, виявлених іншими джерелами	5
Середній час усунення інциденту, хвилин	6
Загальна кількість інцидентів	19
Максимальна тривалість інцидентів, хвилин	40
Максимальна кількість порушених клієнтів	0
Максимальна кількість обірваних з'єднань	500

2) час сповіщення про аномалію. Під цим показником розуміється саме письмове повідомлення заінтересованих осіб. У досліджуваних компаніях з цією метою ефективно використовуються JIRA. Для документування інциденту в GNOC встановлено SLA 2 хвилини, але на практиці це займає не більше 1 хвилини, оскільки багато процесів передачі інформації в ланцюжку Zabbix-JIRA автоматизовані;

3) час відновлення працездатності – це базовий захід оцінки періоду недоступності SaaS для користувачів від моменту часу, коли про це стало відомо, незалежно від способу його відновлення та ступеня автоматизації. Деякі провайдери помилково інтерпретують це як час реакції на інцидент (time to react), проте це некоректно, оскільки ІС недоступна для користувачів доти, доки процедура відновлення не закінчена і QoS не протестовано;

Ці показники обчислюються згідно зі схемою рисунку 2.1 за формулами:

$$MTTD = \sum(T_2 - T_1)\,/N, \qquad (2.1)$$

$$MTTC = \sum(T_3 - T_3)\,/N \tag{2.2}$$

$$MTTR = \sum(T_4 - T_3)\,/N \tag{2.3}$$

де $MTTD$ – середній час виявлення аномалії; $MTTC$ – середній час повідомлення про аномалію; $MTTR$ – середній час відновлення; T_1 – час фактичної події на віддаленому сервері; T_2 – час виявлення аномалії в GNOC; T_3 – час створення документа JIRA; T_4 – час завершення процедури відновлення та переведення інциденту JIRA у статус “Resolved/Closed”; N – кількість інцидентів;

4) час повного циклу зміни/відновлення IC TAT (Turnaround Time). TAT – це категорія світового класу, що використовується для загальної оцінки ефективності заходів щодо технічного обслуговування та експлуатації виробничих систем, незалежно від галузі економіки. У сфері IT TAT означає загальний час обробки заявки на сервісне обслуговування. У повний цикл TAT включений весь процес створення, планування, узгодження змін та їх впровадження у працюючу систему. Стосовно ескалацій, TAT визначається як тривалість повного циклу обробки інциденту від моменту його виникнення в працюючій системі до остаточного відновлення працездатності IC для користувачів.

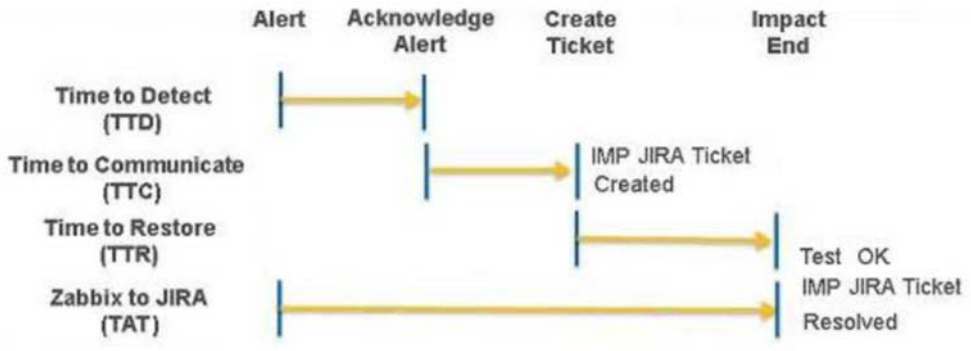

Рис. 2.1. Схема показників інтегральної оцінки якості сервісів

Таким чином, справедливо один із наступних виразів залежно від процедури відновлення вузла:

$$TAT = MTTD + MTTC + MTTR \quad (2.4)$$

$$TAT = MTTD + MTTR \quad (2.5)$$

Вираз (2.5) застосовується за наявності повністю автоматичних процедур виявлення аномалії та відновлення ІВ, коли $\boldsymbol{MTTC}$ і $\boldsymbol{MTTR}$ відбуваються паралельно та ручного втручання у процес ескалації не потрібно.

2.3. Метод динамічного розподілу навантаження на мікросервіс

У досліджуваних великих глобально розподілених хмарних інфраструктурах міжнародних ІТ компанії, основна проблема у підтримці працездатності обчислювальних комплексів пов’язана з динамічним оновленням мікросервісів та необхідністю нульового часу простою, через прискорення циклу розробки нового функціоналу користувача.

За допомогою запропонованого методу динамічного перерозподілу навантаження було вирішено такі завдання:

1) спрощення техніки реконфігурації маршрутизації трафіку:

а) без явного перезавантаження;

б) у єдиній системі.

2) надання базової архітектури та стандартизованої обчислювальної платформи для реалізації парадигми імплементації мікросервісів;

3) надання єдиної методики розгортання обчислювальних схем для розробників ПЗ та/або для обслуговуючого персоналу, що дозволила мати можливість проводити альфа та бета тестування, включаючи експериментальне виробниче тестування шляхом застосування нових або

групових політик, залежно від нових функцій, до окремого облікового запису користувачів або їх невеликої групи;

4) надання співробітникам відділу експлуатації методики управління наступними послугами та діями з нульовим часом простою та без негативного впливу на поточних користувачів інформаційного сервісу:

а) ізоляція – об’єднання апаратної/програмної інфраструктури з індивідуальним обліковим записом;

б) міграція інфраструктури – переміщення послуг між зонами доступності без переналаштування програмних та апаратних компонентів;

в) масштабування комплексу вгору та вниз;

d) перемикання версій ПЗ – переведення сервісів на нову версію програмного забезпечення без явної реконфігурації;

д) встановлення, оновлення, повернення на попередню версію ПЗ;

е) забезпечення еластичності для мікропослуг сервісу.

Мікросервіс є групою так званих програмних контейнерів, розміщених за віртуальною IP-адресою VIP (абр. від англ. Virtual IP), балансувальника навантаження і безпосередньо їх IP-адресою, портом. Контейнерне середовище повинно мати кілька зон доступності. Повинний бути заданий спосіб відокремити або призначити іншу групу облікових записів для явного набору служб, який буде виділено цим обліковим записам.

Конфігурація клієнтської служби має вимагати явної специфікації конкретної служби, найбільш бажане динамічне отримання з конфігурації обчислювального середовища. Необхідно створити обчислювальне середовище, в якому взаємозв'язок між мінімальним налаштовуваним користувачем елементом і елементом мікросервісу, що налаштовується, може встановлюватися динамічно і отримуватися автоматично.

Створення мікросервісу призводить до створення порту VIP та на балансувальнику навантаження, як послуга LBaaS з автоматичним заповненням та/або оновленням запису SRV (абр. від англ. Service Record) в DNS (абр. від англ. Domain Name System) або як послуга DNSaaS (абр. від

англ. DNS as a Service). У маршрутизації додатків для еластичності навантаження на мікросервіс використовується метод Round Robin DNS. Поділ зон може мати фізичну та/або логічну природу, забезпечуючи сумісність з існуючими операційними процесами у обчислювальному комплексі.

Було створено рівень маршрутизації програмних додатків, який є «еластичним рівнем» і запитує глобальний довідник користувачів, щоб знайти вказану службу та направити запит у відповідний сегмент системи для виконання. Еластичність досягається за рахунок використання Round Robin DNS, який гарантує множинні точки входу, не пов'язані з фізичним ресурсом (рисунок 2.2).

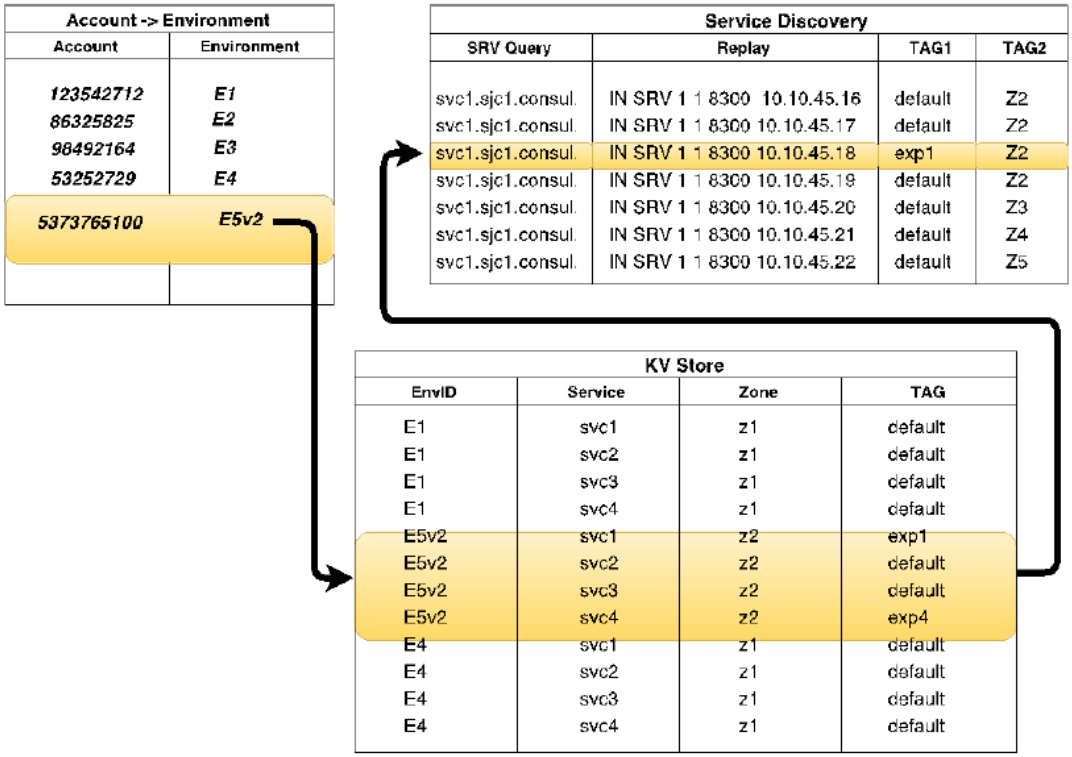

Рис. 2.2. Структура взаємозв'язку логічних обчислювальних середовищ

Для кожного облікового запису створюється зв'язок між обліковим записом користувача та логічним середовищем, що є глобальними даними, логічне середовище при цьому описується унікальним рядком SRV у таблиці маршрутизації. Структура сховища даних ключових значень KV (абр. від

англ. Key Values) визначення середовища визначає взаємозв'язок між ім'ям/ідентифікатором логічного сегмента комплексу і набором послуг, який включає в себе нормальний і повний функціональний набір (перевіряється при створенні облікового запису користувача).

Припустимо, що для створення облікового запису користувача потрібне повне і явне визначення, яке являє собою коротке ім'я служби або повне доменне ім'я FQDN (абр. від англ. Fully Qualified Domain Name), разом з як мінімум 2 додатковими тегами, такими як:

1) Мітка зони – визначає місце розташування служби в зоні, де зона в поточному рішенні відноситься до окремого фізичного екземпляра середовища контейнера, але не обмежується цим. Мітка зони створюється, щоб забезпечити оператору команду для прозорого переміщення служби від одного фізичного об'єкта до іншого, за необхідності;

2) TAG – спеціальний тег, у якому можна визначити мітку, що стосується конкретної версії сервісу. Спочатку він заповнюється за замовчуванням, для простоти його можна опустити.

Виявлення служби, яка інтегрована з СУБД керування конфігурацією CMDB, відіграє роль перетворювача DNS, де всі локальні служби визначені як запис SRV, як показано на рисунку 2.1. Запис заповнюється під час створення служби, але може бути явно змінено за допомогою API. TAG використовуються для визначення специфічних особливостей та функціональних можливостей певного мікросервісу. Зміни TAG також виконуються системою конфігурації через виклик API.

Існує служба API, яка є спільним ресурсом для користувачів із виробничих кластерів. У центрах обробки даних є пул серверів. Ціль – розгорнути нову версію сервера ZAS, яку необхідно включити лише для користувачів, підготовлених у Cluster1, а після закінчення та тестування активних для інших кластерів.

У реалізованому архітектурному рішенні багато кластерної топології розгортання мікросервісів, представленому рисунку 2.3, описані такі компоненти ІС:

1. Рівень маршрутизації додатків – рівень, незалежний та прозорий для сервісів. Виявляє розташування служби на основі інформаційних тегів, які прикріплюються до запиту клієнта і потім переносяться протягом життєвого циклу запиту. Поведінка окремого маршрутизатора виконується за таким алгоритмом:

Крок 1. Якщо запис середовища не існує або термін його дії закінчився, необхідно отримати опис середовища з таблиці структури взаємозв'язку логічних обчислювальних середовищ;

Крок 2. Якщо спроба переслати запит із запису таблиці структури, яка ще не закінчилася, в кінцеву точку не вдалася, за замовчуванням встановлюється час життя TTL (абр. від англ. Time to Live), що викликає перечитування запису з DNS;

Крок 3. Визначити значення TAG із таблиці оточення;

Крок 4. Якщо запис SRV для вхідного запиту із зазначеними тегами вже існує і запис TTL не закінчився, то вхідний запит перенаправляється в кінцеву точку мікросервісу, що існує в записі SRV;

Крок 5. Якщо термін дії запису SRV закінчився або запит надіслано вперше, запит із зазначеними тегами надсилається до таблиці структури;

Крок 6. Якщо відповідь порожня, запит виконується з набором, встановленим за замовчуванням, та пересилається у виявлену кінцеву точку мікросервісу;

Крок 7. Під час початкового завантаження, щоб уникнути проблеми, пов'язаної з недоступністю DNS, таблиця структури за замовчуванням завантажується з конфігураційного файлу завантаження, а потім перевизначається реальною інформацією із запиту DNS.

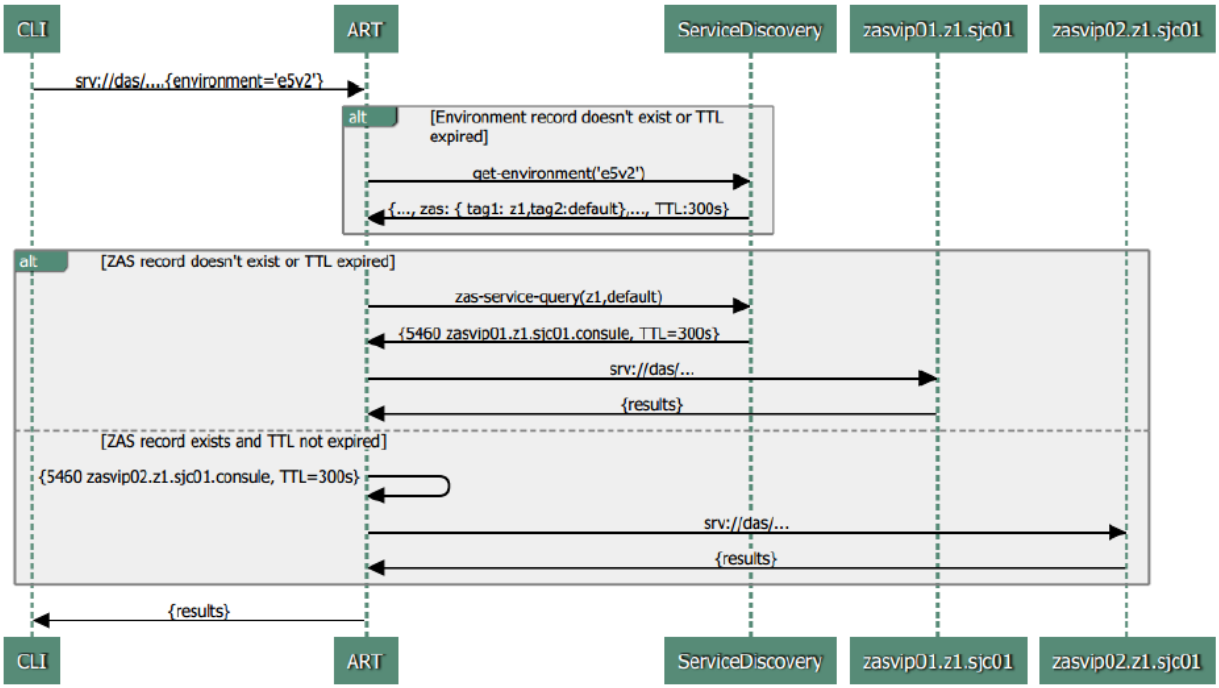

Рис. 2.3. Діаграма кластерного розгортання мікросервісів

2. SDS (абр. від англ. Service Discovery Storage) зберігає таблицю структури взаємозв'язку логічних обчислювальних середовищ і використовується як локальний DNS сервер, сервер маршрутизації додатків працює одним з наступних способів:

а) Підготовлений пошук запитів у такому форматі:

<query or name>.query[.datacenter].<domain>

б) Параметр "OnlyPassing" - управляє поведінкою фільтрації перевірки працездатності запиту. Якщо для цього параметра встановлено значення false, результати включають вузли з перевірками у стані проходження зі станами попереджень. Якщо для цього параметра встановлено значення true, повертаються вузли з перевірками в стані проходження.

Таблиця структури взаємозв'язку логічних обчислювальних середовищ є компонентом середовища і повторно використовується як сховища даних для запису параметрів відношень між обліковими записами та середовищами. Опис середовища та можливих тегів, які пов'язані з різними службами в конкретному середовищі, зберігаються у вигляді пари KV, за умови, що ці

теги витягуються при першому визначенні облікового запису, а потім поширюються разом з будь-яким запитом.

Нижче наведено приклад формату такого запиту (рисунок 2.4):

```
{
   "Name": "zas-service-query",
   "Session": "adf4238a-882b-9ddc-4a9d-5b6758e",
   "Token": "",
   "Service": {
      "Service": "ZAS",
      "Failover": {
         "NearestN": 2,
         "Datacenters": ["sjc01", "iad01"]
      },
      "OnlyPassing": true,
      "Tags": ["z1", "default"]
   },
   "DNS": {
      "TTL": "300s"
   }
}
```

Рис. 2.4. Приклад формування запиту

Вищеописаний метод динамічного перерозподілу навантаження на мікросервіс реалізований та апробований у формі тестового додатку, що інтегрований в глобально розподілену хмарну інфраструктуру із великої кількості віртуальних машин. Внаслідок цього процес автоматичного повторного розгортання програмних служб був прискорений у 3 рази, а безперервне оновлення стало здійснюватися без переривання надання інформаційних послуг для користувачів системи протягом доби та з урахуванням усіх можливих сценаріїв відпрацювання відмови.

2.4. Метод перемикання трафіку в резервну обчислювальну зону

У великій хмарній мультисервісній інфраструктурі, що має велику кількість віддалених VM, встановлюється множина з'єднань між

компонентами спеціалізованих програмних засобів і систем безперервного моніторингу. Відмова однієї VM впливає на інші сервіси при спільної роботі всього обчислювального середовища, що може призвести до перерв в обслуговуванні клієнтів інформаційного сервісу.

На рисунку 2.5 продемонстровано стандартну інфраструктуру SaaS з балансувальником навантаження LBR, де резервне обчислювальне середовище зазвичай резервується в робочій зоні, куди можна перенаправити робоче навантаження. Ця операція називається перемиканням трафіку користувача в резервну зону обчислень.

Кожен обчислювальний регіон складається з 2 і більше центрів обробки даних DC (абр. від англ. Data Center) – основного та резервного. У цій ситуації резервування призначене для перемикання робочого навантаження у разі локального інциденту або повного відключення електропостачання.

Кожен DC складається із ізольованих модулів, які обробляють певну частину даних і трафіку POD (абр. від англ. Point of Data). Основою архітектури POD є СУБД облікових записів ADB (абр. від англ. Account Database) і набір додатків хмарних служб, пов'язаних із загальним сховищем системи повідомлень MSS (Message Store Server) для зберігання запрошень на виклик, запитів до ADB, журналів додатків та подій в інших обчислювальних системах.

Регулярна реплікація даних ADB між контролерами домену дозволяє здійснити швидке перемикання обчислювальних процесів з одного DC в інший у випадку легального інциденту, що відбувся, або з іншої виробничої необхідності.

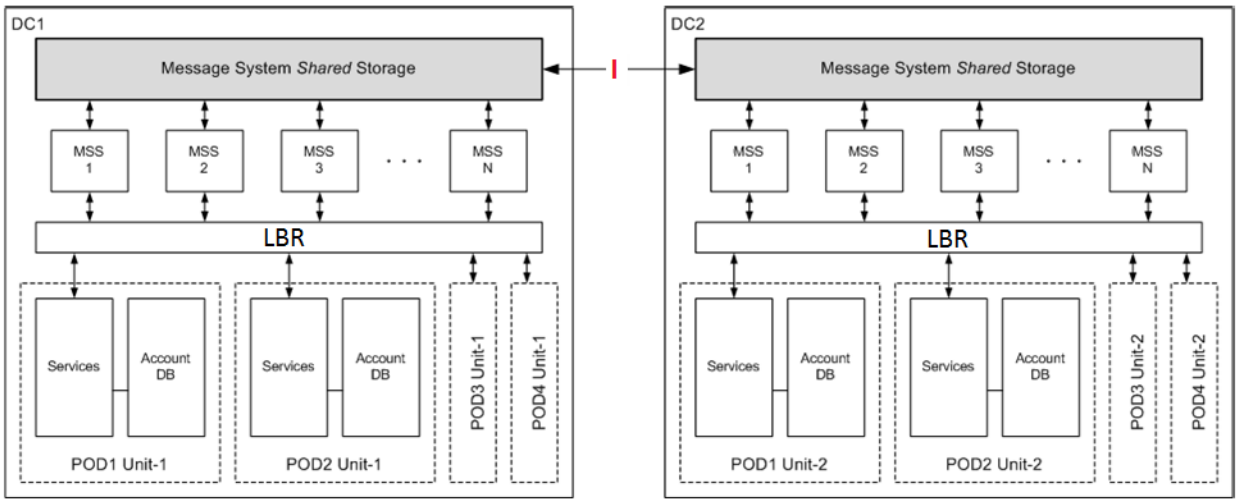

Рис. 2.5. Архітектура SaaS з LBR

Рисунок 2.6 демонструє загальну логіку алгоритму перемикання в резервну зону обчислень. Зважаючи на те, що відмова може статися з будь-яким елементом ІС, включаючи будь-який елемент у резервній обчислювальній зоні, з невідомої причини перенаправлення на пошкоджений вузол не допускається.

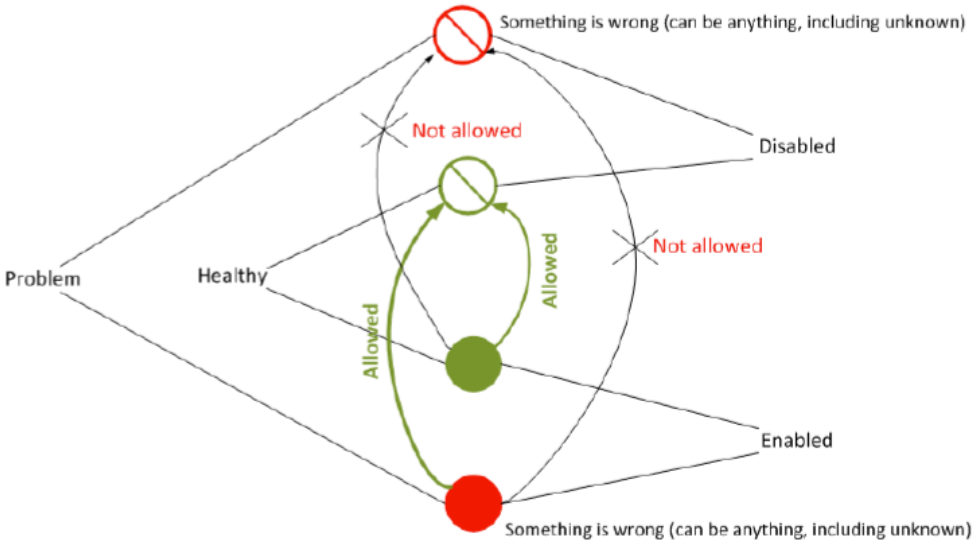

Рис. 2.6. Алгоритм автоматичного перемикання на резервні обчислювальні ресурси ІС

Автоматична процедура перемикання складається з наступних етапів:

Крок 1. Початковий стан перед перемиканням – все робоче навантаження спрямовується в основну обчислювальну зону.

Крок 2. Один сервер із резервного пулу включений в LBR-маршрутизацію та протестований.

Крок 3. Етап 1 повторюється для інших серверів резервної зони.

Крок 4. Як резервна зона повністю активна, інша виключається з маршрутизації та відключається.

Описаний алгоритм передбачає стовідсоткове дублювання обчислювальних ресурсів, оскільки архітектура ІС може мати один набір віртуальних машин, що зарезервовані для багатьох активних наборів, що зменшує надмірність обчислювальної ємності.

Перемикання трафіку в резервну обчислювальну зону виконується швидко і не залежить від кількості віртуальних машин, оскільки кілька етапів на етапі 2 перевіряються паралельно. Наприклад, RNG тривалість автоматичного відновлення 200+ гетерогенних віртуальних машин у зоні становить менше половини хвилини.

Даний підхід застосовується до процедури автоматичного перемикання при збої в працездатності БД, але вимагає більше часу для виконання, оскільки зачіпається реплікація між БД1 і БД2, які, як правило, географічно розподілені і знаходяться в різних центрах опрацювання даних.

Подібне перемикання зазвичай виконується у напівавтоматичному режимі черговою зміною обслуговуючого персоналу, який відповідає за перевірку працездатності SaaS та прийняття рішення на ініціацію автоматичного перемикання всього мережевого трафіку з одного центру до іншого. Для цього в RNG спеціально розроблений графічний інтерфейс, що представляє собою програмний додаток консолі управління потоками користувацького трафіку у всіх географічних зонах ІС, як показано на рисунку 2.7.

Рис. 2.7. Консоль управління всіма потоками трафіку інформаційного сервісу у RNG

Висновки до другого розділу

1. В рамках цього розділу було виконано оцінку продуктивності процесів управління комплексом за запропонованими ключовими показниками динамічного балансування хмарних сервісів. Встановлено надмірність обчислювальних ресурсів у різних часових зонах.

2. Запропоновано метод перенаправлення Інтернет-трафіку в резервну обчислювальну зону на основі розробленого алгоритму автоматичного перемикання.

3. Розроблено інформаційну модель та метод моніторингу стану хмарних сервісів на прикладі Zabbix, які дозволили покращити управління обчислювальними потужностями у хмарній інфраструктурі.

РОЗДІЛ 3
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ АВТОМАТИЧНОГО ВІДНОВЛЕННЯ ІНФОРМАЦІЙНИХ СЕРВІСІВ НА ОСНОВІ ДАНИХ СИСТЕМ МЕРЕЖЕВОГО МОНІТОРИНГУ

3.1. Загальна архітектура системи

В рамках проведеного дослідження описано розроблену розподілену архітектуру для її інтеграції в систему Zabbix, яка була впроваджена для організації системи безперервного моніторингу для забезпечення автоматичного відновлення інформаційних сервісів. Вона є єдиним центром моніторингу віддалених обчислювальних ресурсів у хмарній інфраструктурі та якісно відрізняється кращою візуалізацією подій, що відбуваються в системі.

В рамках даного магістерського дослідження було запропоновано нову архітектуру моніторингу (рис. 3.1), яка складається з веб-сервера Zabbix, сервера БД SQL, ряду проксі-серверів для отримання лічильників від зовнішніх вузлів досліджуваних архітектур з регулярним інтервалом опитування та збереження даних про продуктивність в БД за допомогою окремого Java клієнта.

Існує кілька способів надсилання результатів вимірювань метрик стану компонентів комплексу в Zabbix:

1) Zabbix агент, встановлений на VM та автоматично налаштований для попередньо визначених системних показників VM, таких як завантаження ЦПУ, використання пам'яті, вільне місце на диску, втрата мережевих пакетів, опитування сервера, доступність служби тощо;

2) Zabbix пастка, яка може бути побудована для реалізації будь-якого користувача сценарію на VM і відправки результатів в Zabbix через протокол SNMP (Simple Network Management Protocol);

3) зовнішня перевірка, яка не вимагає установки агента Zabbix і дозволяє виконувати будь-які запити користувача SQL або інший сценарій виконання коду на VM, що повертає результати назад в Zabbix.

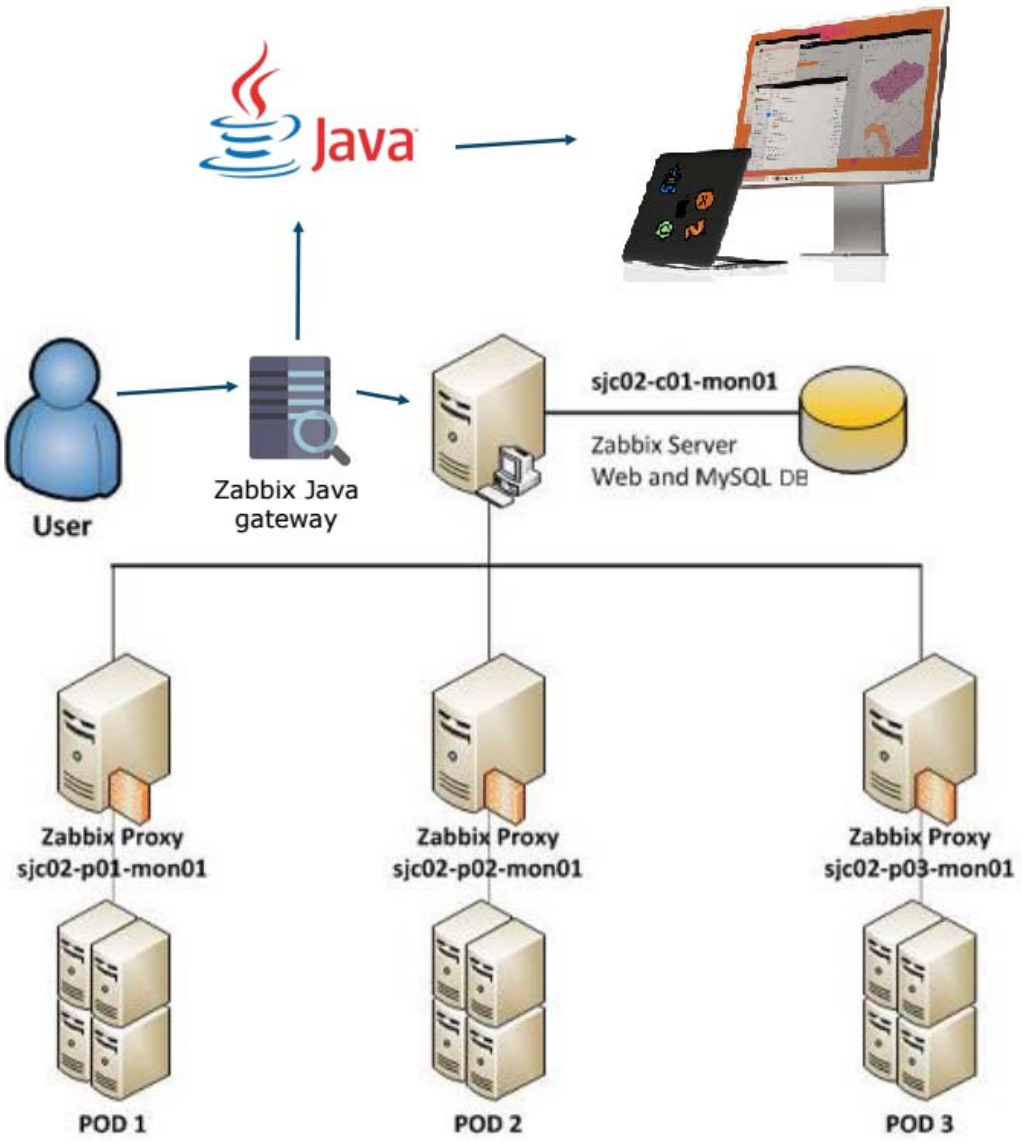

Рис. 3.1. Загальна архітектура системи забезпечення автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу

Події, останні значення метрик та історичні тренди надходять в інтегрованому вигляді до БД Zabbix. Як база даних Zabbix може

використовуватися будь-яка СУБД об'єктно-реляційного типу такі як: MySQL, PostgreSQL, Oracle.

Конфігурація Zabbix включає:

1) список зовнішніх VM, доданих до системи безперервного моніторингу;

2) описи метрик для вимірювання доступності та продуктивності серверів, системних сервісів та бізнес-додатків, що працюють на серверах;

3) тригери, які запускають події, коли перевищено визначені граничні значення допустимих KPIs. Тригери мають різний ступінь важливості події, наприклад: інформаційний, запобіжний, критичний, системний збій;

4) графічні інтерфейси для аналізу даних про продуктивність та історичні тенденції поведінки сервісів.

У рамках магістерського дослідження запропоновано та побудовано нову розподілену архітектуру системи моніторингу Zabbix. Інфраструктури досліджуваних хмарних комплексів є найбільшими з колись відстежуваних Zabbix систем, що складаються з більш ніж 40 тис. VM, що виробляють близько 2 мільйонів метричних значень за хвилину.

Збільшення бази користувача сприяє швидкому збільшенню обсягу мережевого трафіку на ГРВК, що зумовлює формулювання актуального наукового завдання щодо підвищення продуктивності ІС, виключення затримок даних у системі безперервного моніторингу Zabbix для подальшої масштабованості ГРВК і збільшення його пропускної спроможності.

Для вирішення поставленого завдання було вивчено скорочення кількості елементів моніторингу та/або збільшення часових інтервалів опитування. Результат експерименту показав, що така дія потребує великих ресурсно-часових витрат висококваліфікованого персоналу та не призводить до підвищення продуктивності Zabbix.

Додавання більшої кількості серверів, проксі-серверів, високопродуктивних сховищ або іншого обладнання є дорогим рішенням і не має суттєвого економічного ефекту, тому що БД SQL є основним місцем, що

створює ефект росту через чисельність транзакцій читання-запису, що виконуються паралельно.

Як проміжне рішення система Zabbix була розділена на кілька підсистем моніторингу, кожна з яких працює з окремою базою даних. З метою об'єднання розрізнених підсистем у ході експерименту було розроблено Java додаток, що дозволило спростити спостереження за подіями, аварійними сигналами та іншими керуючими даними у комплексі на єдиній графічній консолі моніторингу. Як наступний етап реалізації системи безперервного моніторингу історичні дані були об'єднані для більш ефективного усунення неполадок та аналізу подій.

Наступним етапом стало усунення виявленого ризику затримки даних залежно від поділу між обсягом даних у реальному часі та розміром історичних даних. Наявність у системі моніторингу затримок даних і прогалин у звітності протягом як мінімум 6 хвилин, може, зрештою, призвести до не виявленого збою обслуговування клієнтів, яке може коштувати дуже великих витрат та репутаційних втрат для бізнесу IT компанії.

Для усунення вищезгаданих ризиків було запропоновано та обгрунтовано модифікувати архітектуру Zabbix, яка б дозволила досягти горизонтальної масштабованості із відносно низькими витратами на впровадження. Структурна схема модифікованої архітектури показана на рисунку 3.2. Декілька систем моніторингу Zabbix консолідовано на різних рівнях для задоволення вимог до моніторингу підприємства, включаючи доставку даних без затримок та розривів, підрахунок елементів та тригерів протягом короткого періоду часу, зберігання довгих періодів даних історії, забезпечення подій та видимості даних з усіх логічних та фізичних рівнів. Статистика, обчислювана з отриманих метрик інформаційних сервісів протягом дня зберігається в СУБД MySQL, інші історії та тренди передаються в сховище даних MongoDB, працюючої за принципом NoSQL. Дані в реальному часі не вимагають окремого сховища мережі і можуть

зберігатися на локальному диску або в оперативній пам'яті сервера. Системні події та попередження Zabbix консолідуються на єдиній інформаційній панелі за допомогою веб-API.

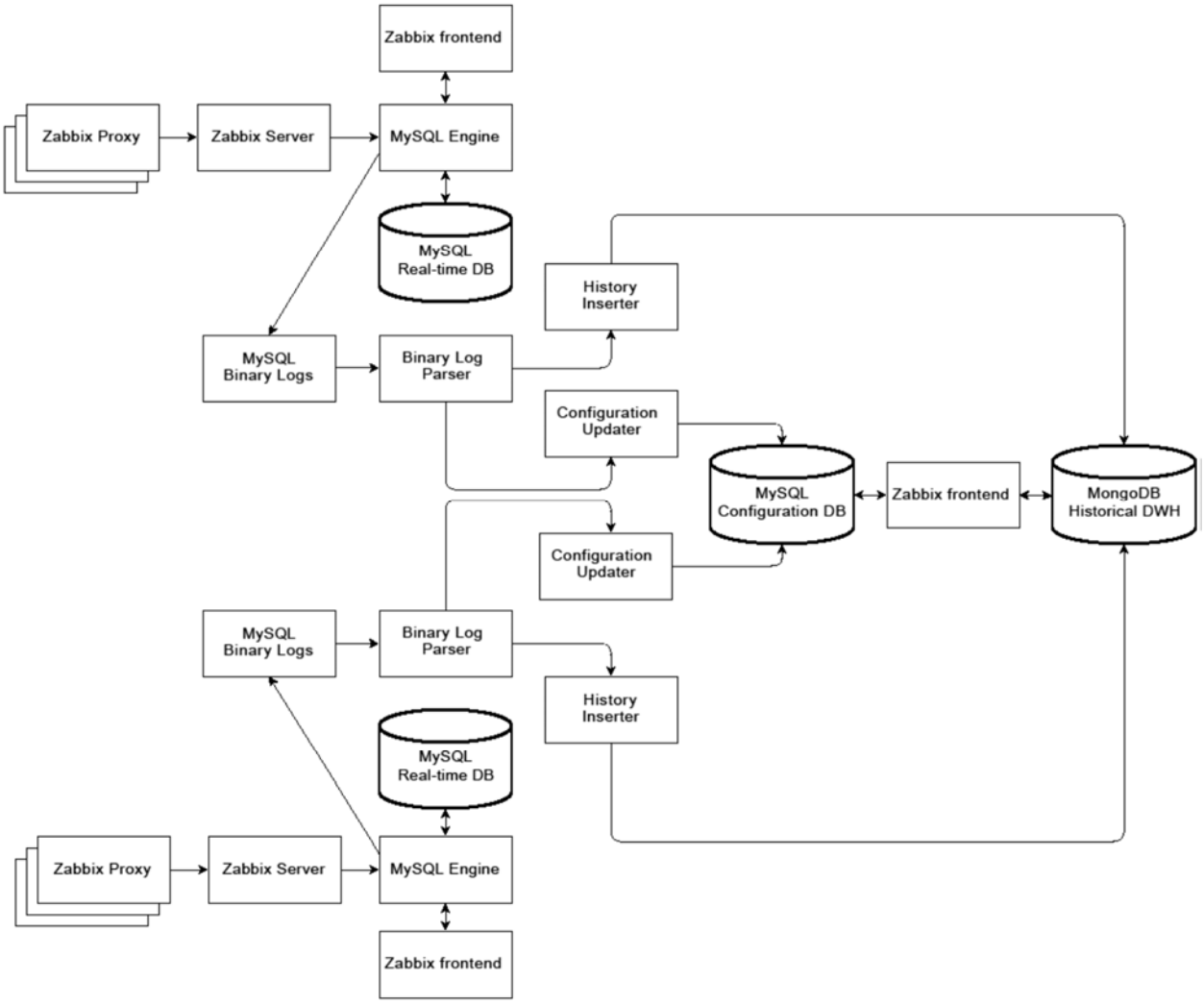

Рис. 3.2. Вдосконалена архітектура для системи моніторингу Zabbix

В результаті проведених досліджень було отримано наступні переваги системи безперервного моніторингу Zabbix:

1) покращено масштабованість системи розподіленого моніторингу, що дозволяє використовувати стільки підсистем моніторингу Zabbix, скільки необхідно для виконання технічних завдань;

2) відсутність затримок у даних моніторингу та прогалин у звітах навіть у піковий час при високому робочому навантаженні;

3) читання та запис в історичній БД були розділені для підвищення продуктивності системи безперервного моніторингу;

4) продовження терміну зберігання історичних даних із 3 до 12 місяців із можливістю нарощування тривалості історичного періоду;

5) запропонована архітектура дозволила здійснювати налаштування СУБД, що підвищило продуктивність БД без затримки передачі у кожному конкретному випадку реалізації;

8) розподілена архітектура дозволила використовувати Zabbix незалежно від порога масштабованості системи з точки зору серверів та показників моніторингу;

9) для розподіленої бази даних Zabbix було запропоновано створення інтегрованої панелі моніторингу, щоб об'єднати дані моніторингу з окремих джерел на єдиний графічний інтерфейс безперервного моніторингу з доступністю 24x7.

3.2. Структура підсистеми зберігання інформації

В рамках реалізації системи розроблена та реалізована СУБД системи автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу на основі єдиної інтегрованої конфігураційної бази даних CMDB в архітектурі храмних центрів опрацювання даних. За основу та взяти поточні бізнес-процеси та програмні засоби досліджуваних програмних комплексів.

Компоненти системи складаються з наступних ідентифікаторів:

1) Rackspace – фізичне розміщення стоків серверного обладнання центру опрацювання даних;

2) Об’єкти – опис існуючих серверів програмних додатків, включаючи конфігураційні параметри;

3) простір IPv4 – база даних IPv4-адрес, їх розподіл і перерозподіл по серверах;

4) простір IPv6 – база даних IPv6-адрес, їх розподіл і перерозподіл по серверах;

5) Файли – сховище спеціалізованих файлів;

6) Звіти – підсистема формування звітів з бази даних CMDB;

7) IPv4 SLB – база даних конфігураційних налаштувань балансувальників навантаження;

8) 802.1Q – віртуальні локальні мережі VLAN (абр. від англ. Virtual Local Area Network).

Розроблена СУБД дозволяє управляти такими процесами постійного моніторингу в програмному комплексі по новому:

1) додавання та видалення обчислювальних ресурсів у реальному часі, виходячи з пікового навантаження на інформаційні сервіси;

2) встановлення та оновлення системного та спеціального програмного забезпечення;

3) облік та мінімізація затрат на обслуговування вичислювальних ресурсів центрів опрацювання даних.

Розроблену систему можна застосовувати як для відкритих, так і власних центрів опрацювання даних компаній. Порівняльний аналіз і урахування витрат на обслуговування інфраструктури досліджуваних IT-компаній підтвердив, що більш ефективно використовувати обчислювальні ресурси відкритих хмарних центрів.

Для вирішення поставлених завдань була розроблена інформаційна модель інтеграції системи моніторингу з використанням інструментальних засобів моделювання СУБД Gliffy (рисунок 3.3).

На відміну від існуючих рішень, запропонована нова система критеріїв і структура даних описує зовнішню компоненту IC разом з використанням критерію System Unit, який об’єднує обчислювальні компоненти та вузли IC за набором унікальних параметрів, властивих тільки цьому типу. Крім того, для кожного обчислювального вузла було запропоновано включити додатковий критерій System Relation, що дозволить об'єднати будь-які два

вузли, з точки зору їх системних зв'язків, і далі реалізувати класи реляційних зв'язків за логікою наслідування, враховуючи всі види фізичних і логічних мережевих інтерфейсів і з'єднань, а також транзит мережевого трафіку.

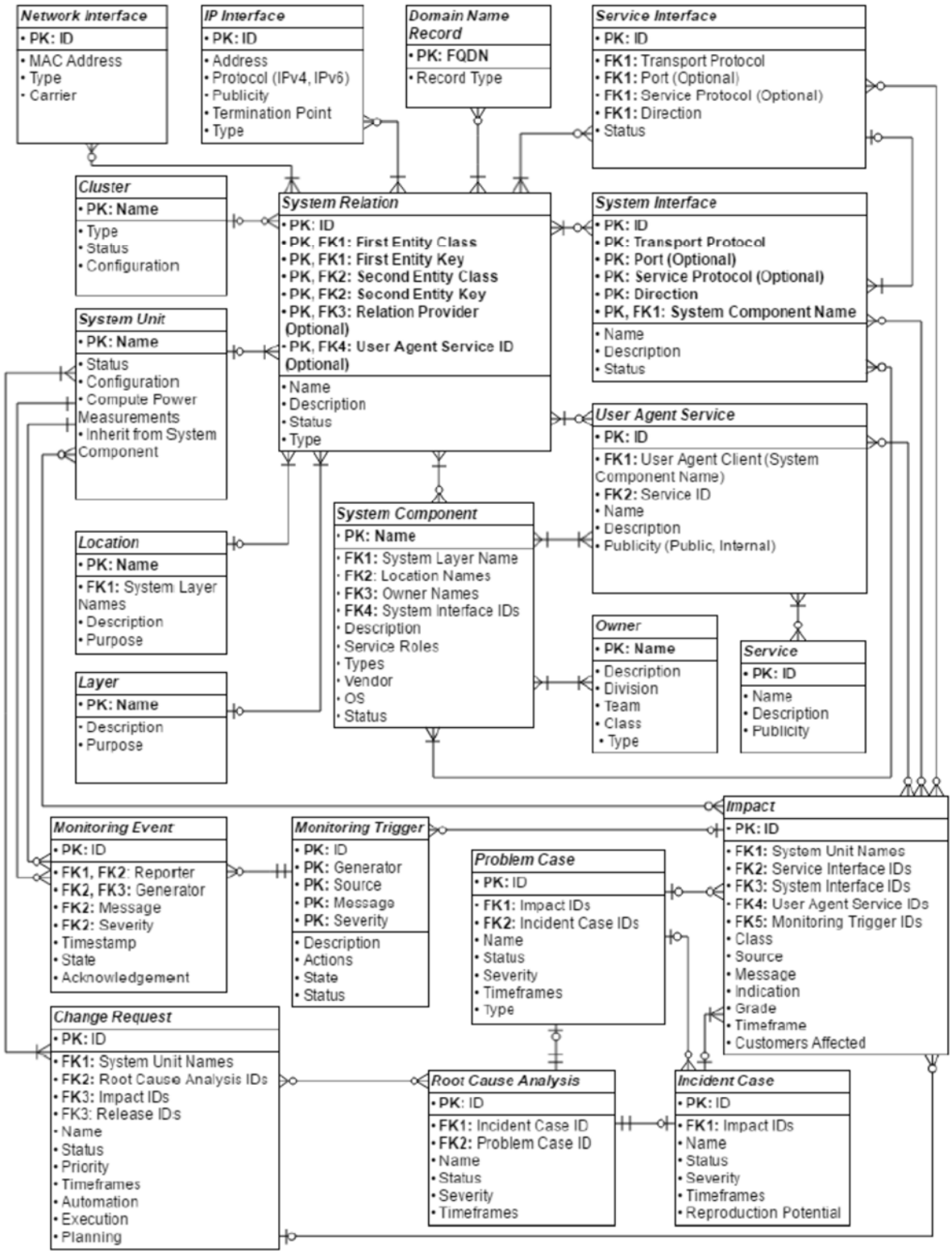

Рис. 3.3. ER діаграма бази даних

Для опису «базового вузла» всіх вичислювальних компонентів і мережевих вузлів був наведений критерій System Component, який включає в себе верхньо-рівневий набір параметрів вичислювальних компонентів і мережевих вузлів.

При цьому Системний Компонент може бути представлений у кількох розташуваннях (перевод з англ. Location) і являє собою унікальний системно-вимірюваний рівень, який в програмному комплексі може бути використаний для виконання різних обчислень, таких як: вичислювальні середовища для надання IT-послуг клієнтам компаній або лабораторне середовище для розробників ПЗ і тестувальників.

Для кожної системної одиниці були введені наступні ідентифікатори, що дозволяють об’єднати моніторингові події за належністю:

• Сервіс User Agent (користувацький сервіс).

• User Agent Client (користувацький клієнт).

• Власник (власник).

• Системний інтерфейс.

• Мережевий інтерфейс.

• Запис доменного імені.

• Кластер.

• Сервіс.

У разі виникнення відмови в програмному комплексі, моніторингові заходи можуть бути також згруповані за результатами оперативного технічного розслідування причини його виникнення. Для такого сценарію були запропоновані та реалізовані наступні ідентифікатори:

• Тригер моніторингу.

• Проблемний випадок.

• Impact (негативний вплив)

• Incident Case (випадок інцидента).

• Change Request (запит на зміну).

Вплив ідентифікується за користувацькими запитами. Вплив має часовий діапазон і налічує такі градації: переривання, скорочення якості, переривання сесій, втрата надмірності, втрата ємності. Вплив торкається або не торкається користувачів. Якщо вплив не впливає на користувача, він розцінюється як потенційний і належить Problem Case. Якщо вплив впливає на користувачів, він належить Incident Case. Якщо за результатами розслідування причин RCA інцидент має ймовірність повторення, він має стати частиною проблеми.

Знайдена проблема працездатності програмного комплексу супроводжується технічним розслідуванням причин її виникнення. Результатом розслідування причин інциденту є екстрений Change Request. Результатом розслідування причин проблеми може бути як екстрений, і запланований запит на зміну. В рамках запиту на зміну відбувається оновлення конфігурації системних компонентів. Запит на зміну може бути викликаний результатом розслідування причин відмови працездатності ІС або інциденту. Незалежно від причин провадження змін у системі, вони здатні викликати як позитивний, так і негативний вплив на працездатність системи в цілому.

3.3. Модель бази даних для зберігання програмних конфігурацій компонентів і сервісів

Розвиток інформаційних систем та технологій призводить до необхідності використання набору обчислювальних елементів для побудови структури севісно-орієнтованого центру. Незважаючи на географічний поділ єдиного інформаційного середовища, ІТ компанії потребують централізованого управління технічним обслуговуванням центрів з урахуванням постійних змін у топології, пов'язаних з модернізацією

апаратних обчислювальних компонентів, безперервним варіюванням кількості віртуальних машин у хмарі.

Наявність бази даних управління конфігураціями CMDB допомагає підвищити точність конфігураційних відомостей CI, що зберігаються (Configuration Item) про поточний стан обчислювальних компонентів в ІС. Збір, погодження та обслуговування CI роблять дані надійними та придатними для використання в системах автоматизованого технічного обслуговування.

Записи взаємозв'язків сервісів один з одним, незалежно від того, знаходяться вони між структурами конфігурації служб або всередині структур конфігурації служб, зберігаються в ядрі CMDB, що визначає відмінність CMDB від бази даних зберігання параметрів та ідентифікаторів серверів програмного комплексу, оскільки системи управління відповідають за зберігання інформації про обчислювальні елементи, а не за конфігураційні параметри, що описують ланцюжки логічної залежності між цими елементами. Без розуміння взаємозв'язків неможливо зрозуміти конфігурацію обчислювального середовища або як об'єднані компоненти надають та підтримують інформаційні послуги.

Використання CMDB дозволяє налагодити методику для безперервного покращення існуючих робочих процесів (рисунок 3.4), надавати нові можливості для автоматизації технічного обслуговування сервісів, які раніше були недоступні при використанні різнорідних сховищ даних.

Розроблена інформаційна модель інтеграції процесів експлуатації хмарних сервісів реалізована на практиці, де реалізовано такі підходи:

1) використання єдиного механізму авторизації SSO (Single Sign on);

2) мікросервісна архітектура віртуальних серверів;

3) використання шини передачі повідомлень за принципом оповіщення клієнтів для обміну даними між мікросервісами;

4) доступ до даних через абстракцію API.

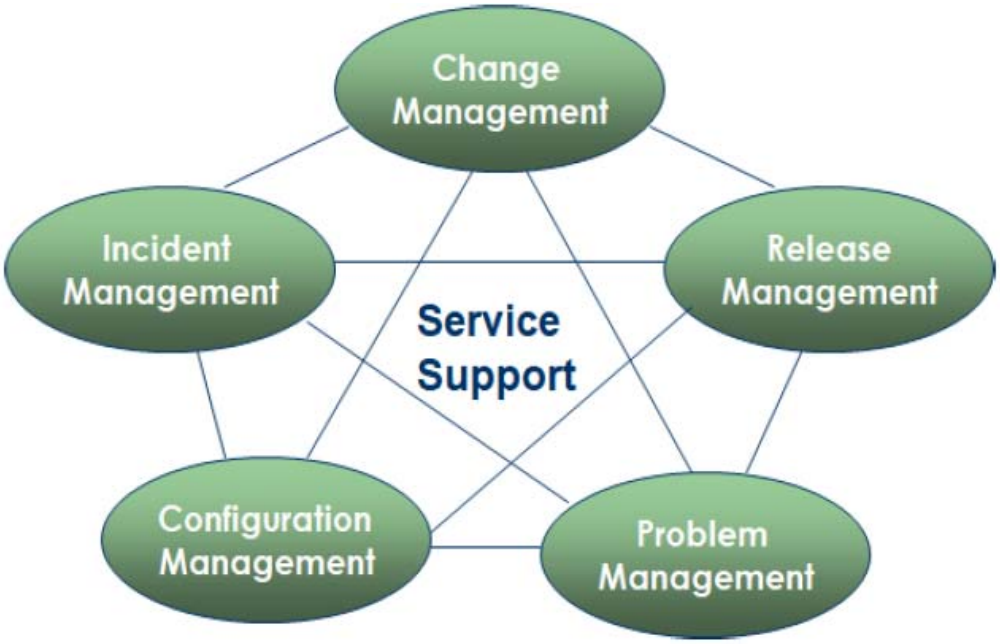

Рис. 3.4. Робочі процеси обслуговування програмно-апаратного комплексу

Головними перевагами наявності CMDB є:

1) Консолідація сервісів, які раніше були розподілені у різних службових підсистемах;

2) Єдина база достовірних даних;

3) Об'єднання всіх служб/джерел та споживачів інформації.

Це призводить до зниження можливості автоматизації робочих процесів з технічної підтримки, що збільшує витрати компанії на обслуговування комплексу.

З метою вибору відповідної технології для створення CMDB були визначені основні статуси у життєвому циклі зберігання сервісів для будь-яких компонентів хмари (таблиця 3.1), які можуть бути фізичними, віртуальними або контейнерними.

Для вирішення задачі раціонального вибору архітектури CMDB було здійснено розрахунок потоків вимог на обслуговування від компонент ІС, що надходять до бази даних управління конфігураціями та виходять з неї, тривалості очікування та довжини черг з урахуванням періодів опитування

різнорідних джерел зберігання сервісів, визначених бізнес-завданнями та представленими в таблиці 3.2.

Таблиця 3.1

Атрибути життєвого циклу зберігання сервісної інформації

№	Атрибут	Опис
1	Incomplete	Компонент системи не готовий до використання
2	New	Компонент створений і готовий до інсталяції та подальшої конфігурації в ІС
3	Unallocated/Cache	Компонент пройшов процес ініціалізації та готовий до додавання до обчислювального пулу.
4	Provisioning (HW)	Розпочато підготовку апаратної частини
5	Allocated	Компонент перебуває у робочому стані у обчислювальному пулі
6	PREP (Application provisioning)	ПЗ програми знаходиться в процесі встановлення та налаштування
7	LIVE	Компонент обслуговує трафік користувача
8	Remediation	Компонент знаходиться в стані виправлення неполадок, що виникли.
9	Cancelled	Компонент більше не потрібен і чекає на виведення з експлуатації
10	Decommissioned	Компонент було видалено з системи і більше не може керуватися
11	Debug	Вказує, що компонент знаходиться в режимі налагодження та показники / оповіщення у системі моніторингу з цього сервера можуть ігноруватися
12	Maintenance	Компонент піддається деякому технічному обслуговуванню та не повинен розглядатися для виробничого використання

Головним показником продуктивності системи став час очікування запису чи читання всього спектра сервісів з урахуванням кількості обчислювальних компонентів.

Таблиця 3.2

Атрибути періодів опитування джерел зберігання сервісної інформації

№	Тип даних	Джерело інформації та період опитування
1	Фізичні сервери	RackTables – щогодини
2	Мережеві підключення	RackTables – щогодини
3	Віртуальні сервери	RackTables – щогодини
4	Тип віртуальних машин на одному фізичному сервері	vCenter – двічі на день
5	IP адреси	RackTables – щогодини
6	Віртуальні пули серверів	RackTables – щодня
7	Логічні IT середовища	ADS – щодня, Amazon – щодня
8	Обслуговування інформаційних сервісів	CMP – миттєво, Zabbix – щогодини
9	Інциденти	IMP – миттєво, Zabbix – миттєво

Результати розрахунку показали, що наявність єдиної CMDB здатна обслуговувати понад 10 млн. конфігцрацій згідно з вимогами щодо забезпечення заданих періодів опитування різнорідних систем управління працездатністю комплексу.

Для практичної реалізації CMDB з урахуванням необхідності надання графічного представлення взаємозв'язків обчислювальних елементів в була обрана СУБД, побудована за технологією Neo4j, заснована на графовій системі з відкритим вихідним кодом, реалізованою мовою програмування Java. Стандартний алгоритм послідовності створення компоненти з погляду

життєвого циклу зберігання конфігурації в Configuration Management Database представлений рисунку 3.5.

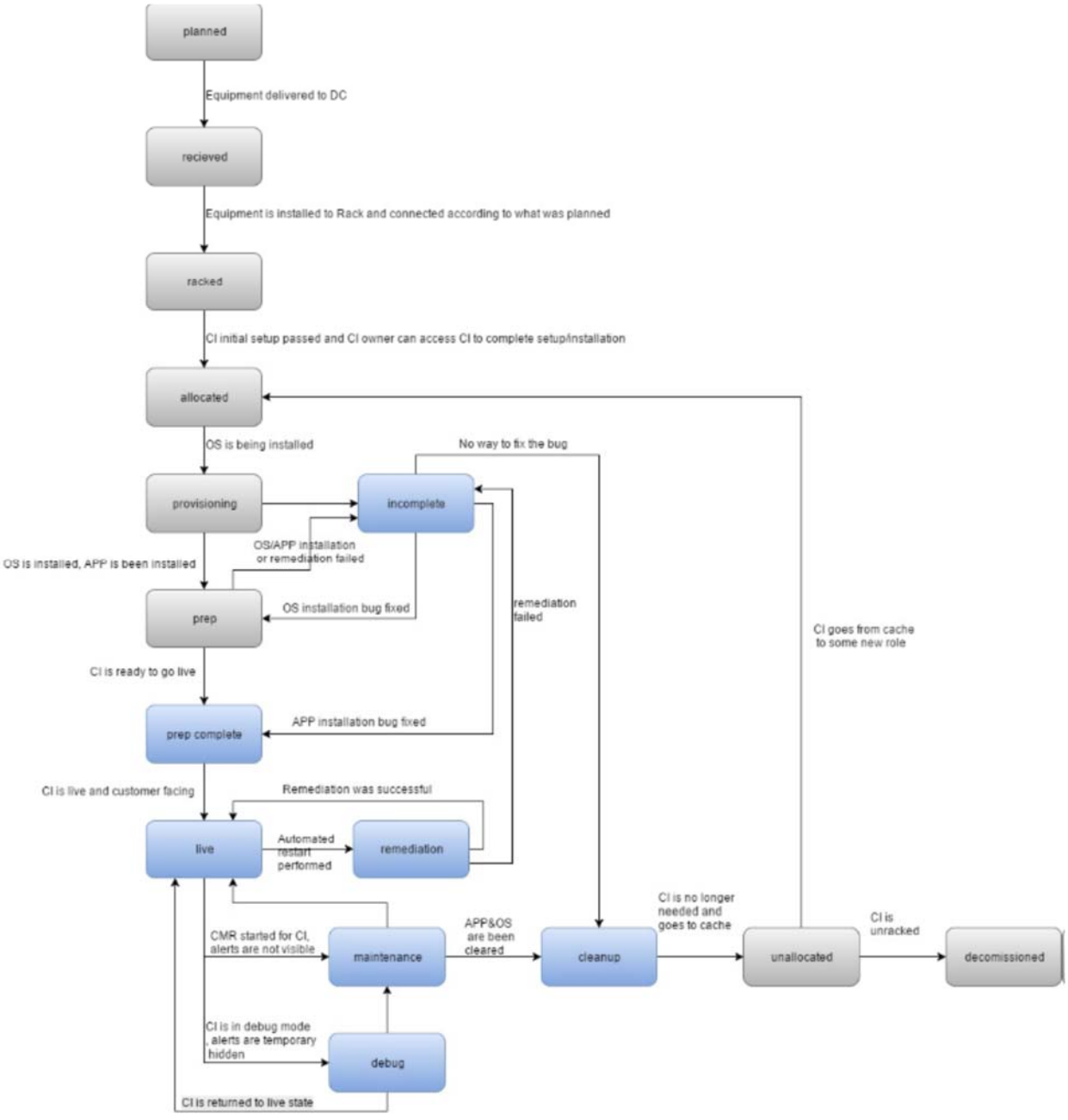

Рис. 3.5. Схема реалізації процедури зберігання інформації при аналізі стану інформаційних сервісів на основі даних систем мережевого моніторингу

3.4. Особливості програмної реалізації Java додатку для автоматичного відновлення інформаційних сервісів

В рамках виконання магістерської роботи реалізований Java додаток, який інтегрується в загальну архітектуру програмно-апаратного комплексу надання сервісних послуг з використанням хмарних технологій, і який використовується для отримання та опрацювання даних з метою автоматичного відновлення інформаційних сервісів на основі даних системи мережевого моніторингу Zabbix.

На рисунку 3.6 представлено головне вікно додатку для автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

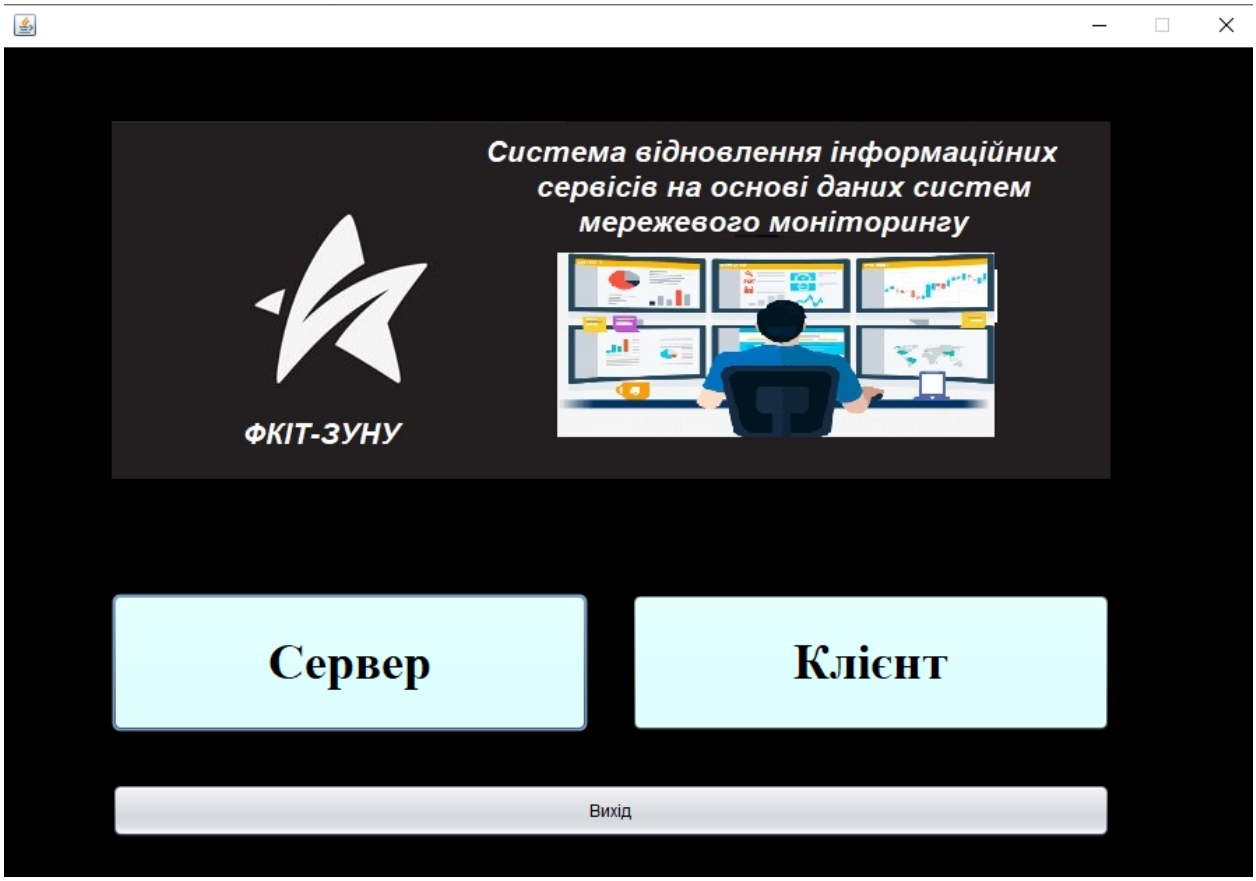

Рис. 3.6. Головне вікно додатку для автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу

Додаток працює в двох функціональних режимах – як серверна частина так і клієнтська частина. Для під’єднання до серверу необхідно вказати порт для підключення до системи моніторингу (рисунок 3.7).

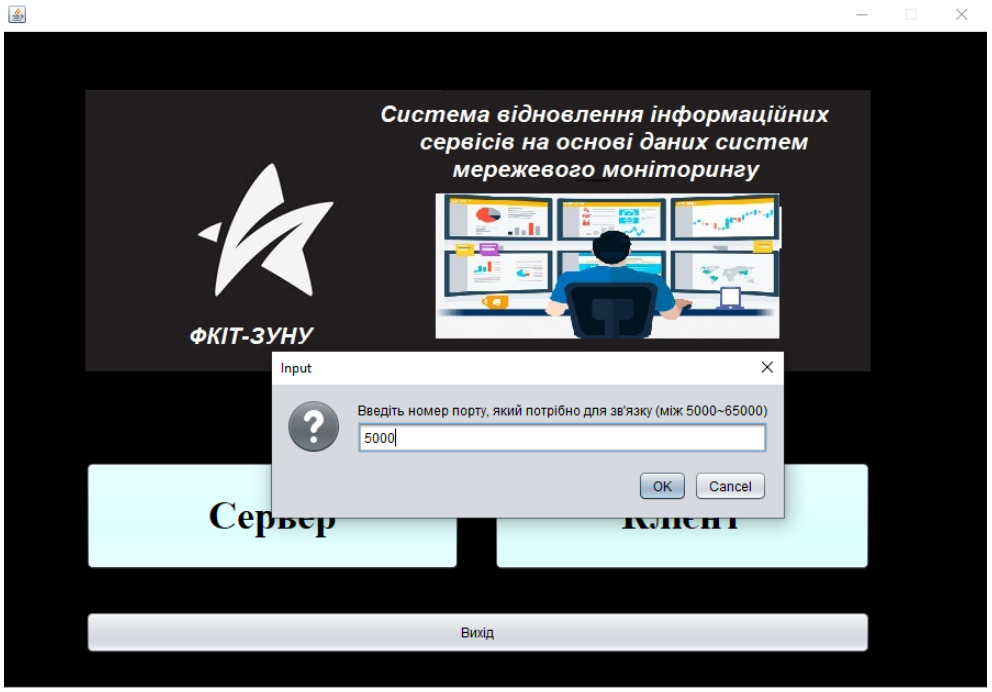

Рис. 3.7. Вікно під’єднання до сервера

На рисунку 3.8. представлено копію вікна із основними атрибутами для встановлення з’єднання та переходу в серверну частину додатку.

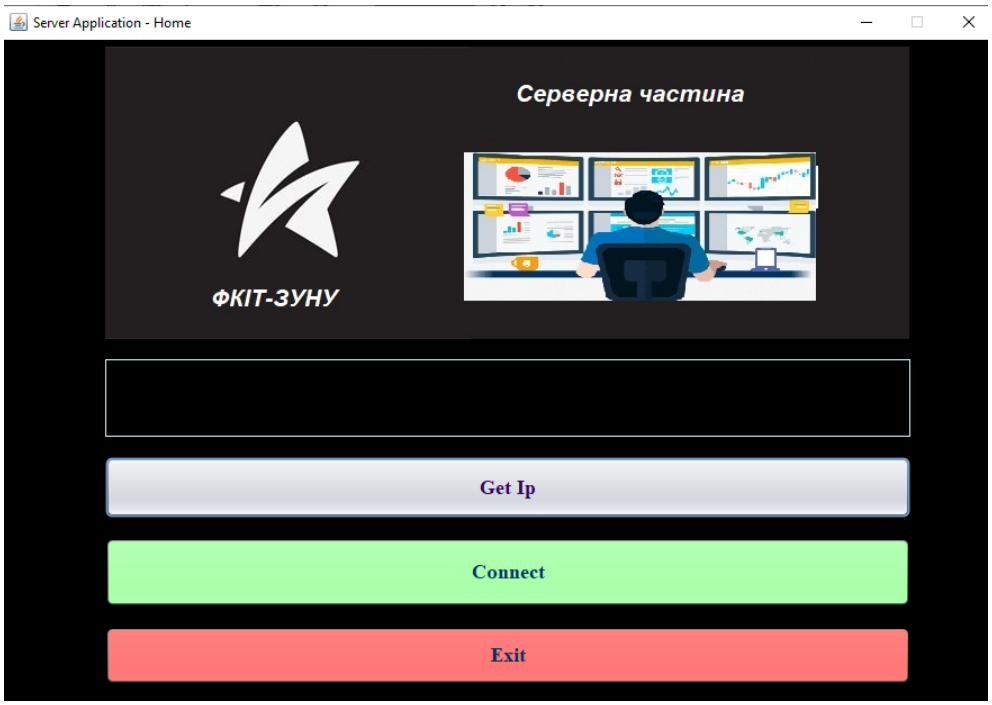

Рис. 3.8. Серверна частина додатку

На рисунку 3.9 представлено клієнтську частину розробленого додатку в рамках запропонованої архітектури.

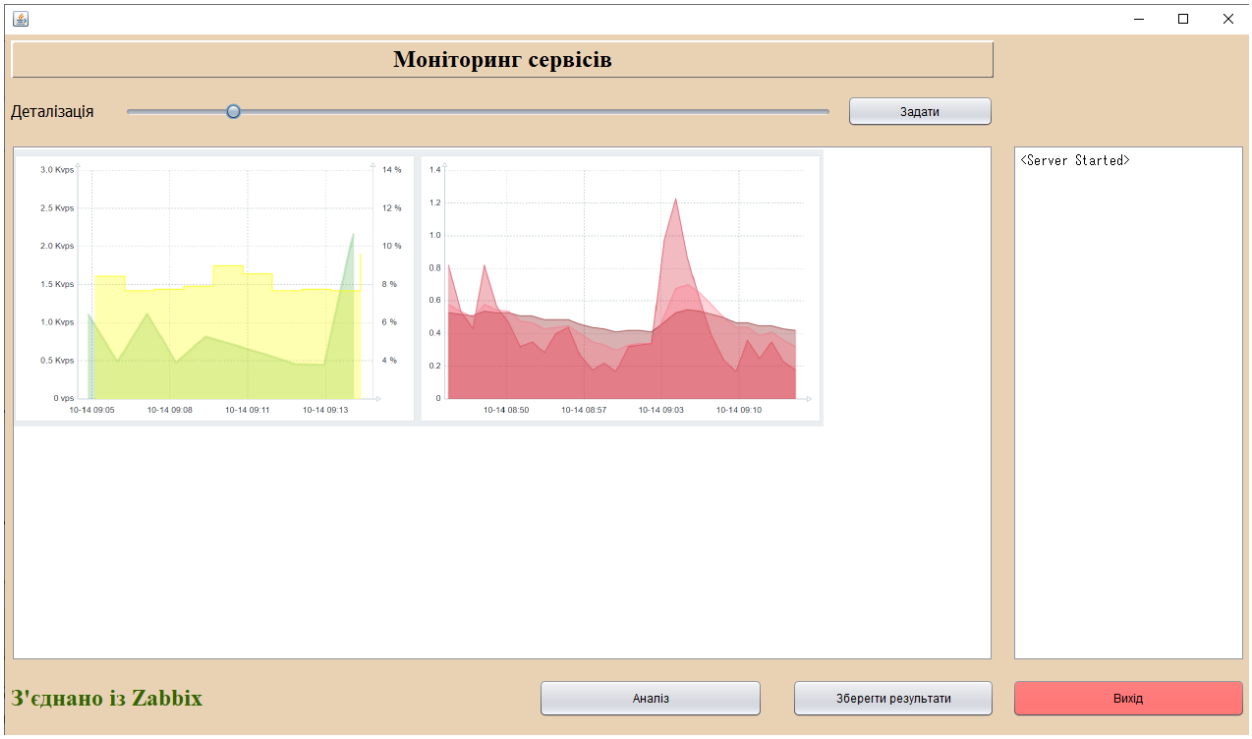

Рис. 3.9. Клієнтська частина додатку автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторинг

Незважаючи на простий інтерфейс, запропонований додаток дозволяє швидко виявляти збої на вузлах, використовуючи підключення до системи Zabbix та програмно реалізовані в 2 розділі роботи методи та алгоритми

3.5. Експериментальні дослідження

Як приклад, розглянемо процесинговий вузол, що складається з чотирьох спеціальних обчислювальних компонентів (рис. 3.10), в якому один знаходиться в стані технологічної відмови, проте при цьому зберігається певний обсяг ємності вузла так як кількість працездатних компонентів, що залишилася, здатна обслужити користувацькі запити.

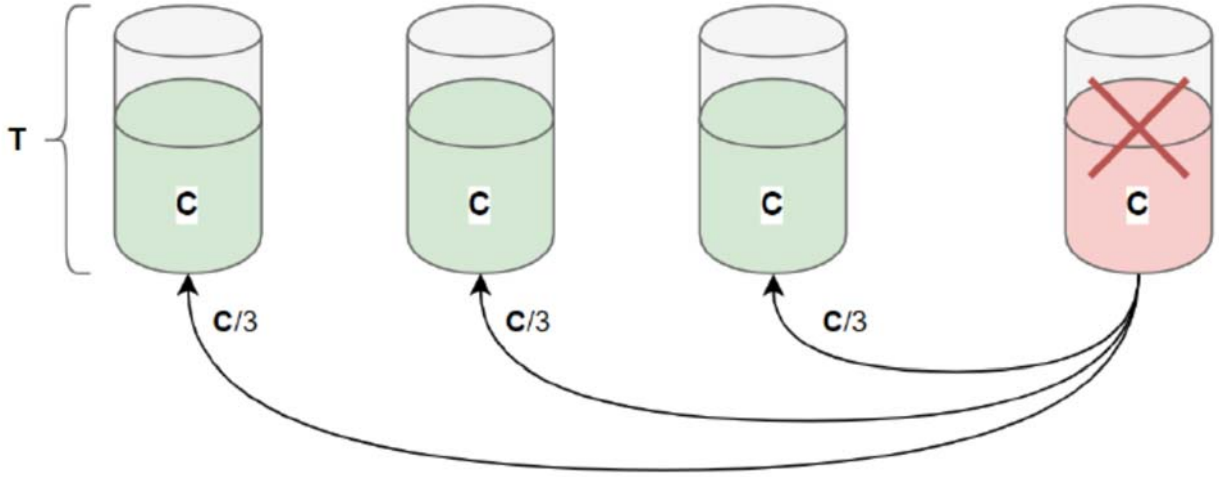

Рис. 3.10. Процесинговий вузол, що складається з чотирьох компонент

Враховуючи, описаний в другому розділі підхід, отримуємо математичний вираз нерівності (3.1) та результат обчислення (3.2):

$$\mathrm{C} + \mathrm{C}/3 < T \tag{3.1}$$

$$\mathrm{C} + 75\% * T \tag{3.2}$$

де T — це загальна доступна кількість процесингового ресурсу на кожному вузлі; C — це кількість ресурсу, що використовується в даний час. При чому C не повинен перевищувати деякого порогового значення, щоб тримати процесинговий ресурс у резерві у випадку, якщо один із компонент вийде з ладу і навантаження перерозподілиться між тими, що залишилися в робочому стані.

Отже, якщо один із компонент виходить з ладу, то робочі додатково отримають навантаження $V(3)$:

$$V = C/n\,, \tag{3.3}$$

де n - кількість компонент у конкретному процесинговому вузлі.

Отже, на кожному вузлі доступне обчислювальне навантаження та пропускна здатність не повинна перевищувати 75% від максимально доступної ємності для того, щоб мати можливість обробляти виклики в процесинговому вузлі, без деградації якості обслуговування у випадку, якщо один із вузлів у групі буде недоступним. Для цього, в ході експерименту, були задані порогові значення для генерації повідомлень у системі моніторингу: критичне (Critical) – 75%, та попередження (Warning) – 60% (рис. 3.11).

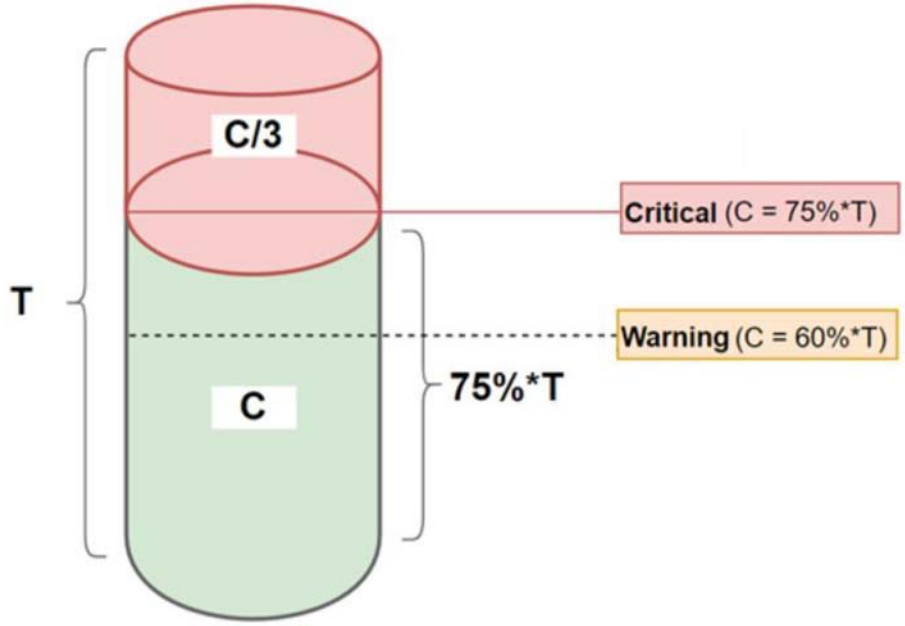

Рис. 3.11. Робоча модель ємності вузла

Використовуючи Zabbix, запишемо вираз обчислення використання ресурсів та роль процесора трафіку пакетів RTP у відсотковому співвідношенні:

{
"denominator": <digit value>,
"formula": "usage",
"metric_key": <calculated metric key>,
"metric_name": <calculated metric name>,
"trigger": [
{
severity: "warning",
threshold: "C-15%"

```
}
{
severity: "critical",
threshold: "C"
}
]
"terms": [
{
"host": <RTP>,
"metric": "webRTC[<RTP term metric>]"
}
]
}
```

Далі необхідно створити методику моніторингу для схеми об'єднання процесингових вузлів в єдиний комплекс з урахуванням ємності каналів передачі пакетів на TG (Trunk Group), що представлено на рисунку 3.12.

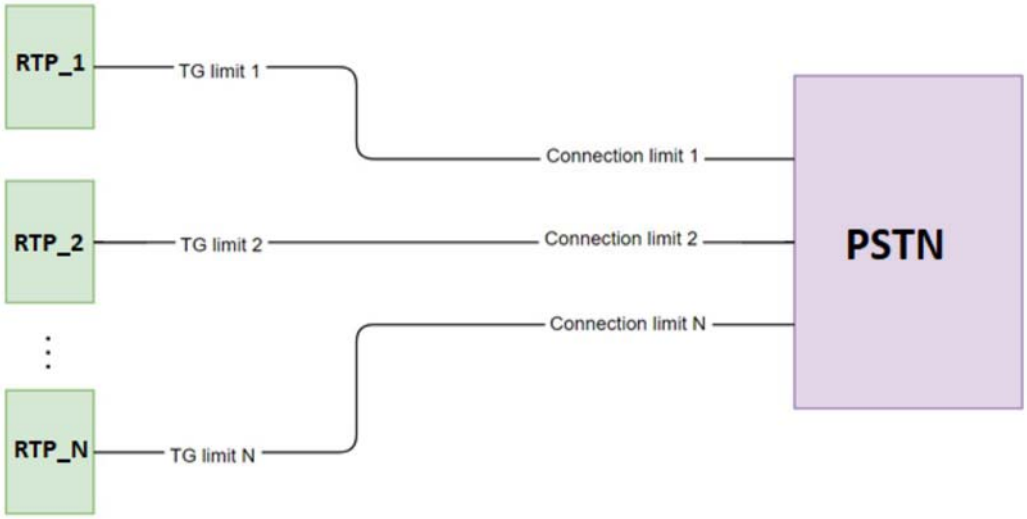

Рис. 3.12. Схема об’єднання вузлів

У цьому випадку в комплексі усі групи з'єднувальних ліній на боці процесингових вузлів RTP_n мають таке ж обмеження одночасних викликів, яке має на своєму боці PSTN (Public Switched Telephone Network). Найчастіше у комплексі є обмеження кількість одночасних викликів (Connection Limit) для груп з'єднувальних ліній для конкретної TG оператора

зв'язку у конкретному регіоні. Відповідно, для моніторингу ємності вузлів RTP_n для відстеження лімітів трафіку пакетів в одній TG було запропоновано використовувати наступний вираз Zabbix:

```
{
terms: [
{
host: <RTP>,
metric: "webRTC[<Trunk Group metric of outbound/inbound calls usage>]",
}
]
metric_name: <calculated metric name>,
metric_key: <calculated metric key>,
formula: "usage",
denominator: {
host: <RTP>,
metric: "webRTC[<Trunk Group metric of totals calls configured>]",
}
trigger: [
{
severity: "critical",
threshold: "X",
}
{
severity: "warning",
threshold: "0,8*X",
}
]
}
```

Трафік, що надходить від операторів PSTN розподіляється в комплексі між двома підмножинами процесингових вузлів RTP_n та відповідними TG, що використовуються для обробки трафіку пакетів (рисунок 3.13).

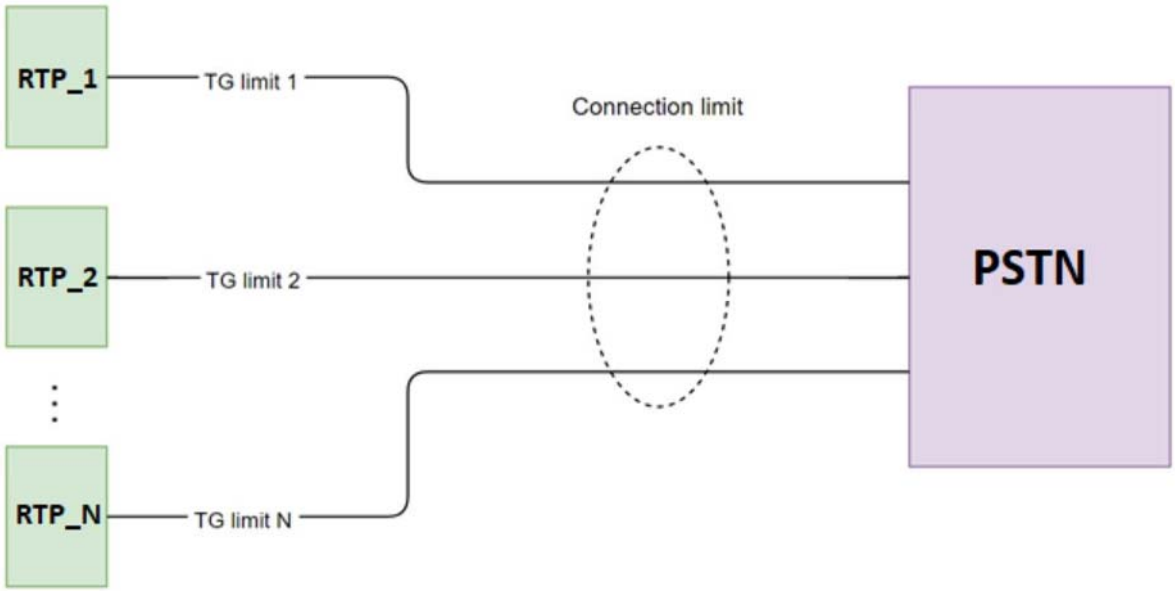

Рис. 3.13. Робоча схема приєднання вузлів з врахуванням лімітів

Для здійснення моніторингу повного завантаження з'єднувальних ліній оператора PSTN, було запропоновано за допомогою виразу Zabbix порівняти загальну кількість одночасних викликів відповідних груп з'єднувальних ліній до вузлів RTP_n з лімітом з'єднувальних ліній оператора зв'язку для того, щоб обчислити відсоток використання групи з'єднувальних ліній від оператора:

{
"metric_name": <calculated metric name>,
"metric_key": <calculated metric key>,
"formula": "comparison"
"terms": [
197
{
"host": <RTP_1>,
"metric": "webRTC[<RTP_1 API term metric>]"
},
{
"host": < RTP_1>,
"metric": "webRTC[<RTP_2 API term metric>]"
},
{
"host": <SBC N>,
"metric": "webRTC[<RTP_N API term metric>]"

```
}
]
"denominator": {
"host": <RTP_1>,
"metric": "webRTC[<RTP_1 API denominator metric>]"
}
}
```

Запропонований метод безперервного моніторингу з використанням розробленого методу динамічного перерозподілу обчислювальних ресурсів між вузлами став ефективним способом покращення продуктивності та управління обчислювальними потужностями у великій хмарній сервісній інфраструктурі.

Висновки до третього розділу

1. Наведено обгрунтування вибору технології реалізації програмного програмне забезпечення автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

2. Розроблено інформаційну модель та метод моніторингу стану вузлів сервісно-орієнтованої системи на прикладі Zabbix, яка дозволили покращити управління обчислювальними потужностями у хмарній інфраструктурі.

ВИСНОВКИ

В результаті виконання магістерської роботи отримано наступні теоретичні та практичні результати, які грунтуються на розробці методів та засобів автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

Відповідно до поставленої мети у роботі отримано наступні результати:

1. Виконано аналіз сучасних підходів до організації хмарних обчислень, та розкрито суть організації спеціалізованих прогшрамних засобів на їх основі.

2. Здійснено оцінку продуктивності процесів управління комплексом за запропонованими ключовими показниками динамічного балансування хмарних сервісів

3. Розроблено новий метод перенаправлення Інтернет-трафіку в резервну обчислювальну зону на основі розробленого алгоритму автоматичного перемикання.

4. Реалізовано інформаційну модель та метод моніторингу стану хмарних сервісів на прикладі Zabbix, який дозволили покращити управління обчислювальними потужностями у хмарній інфраструктурі.

5. Розроблено програмне забезпечення для автоматичного відновлення інформаційних сервісів на основі даних систем мережевого моніторингу.

6. Проведено експериментальні дослідження розроблених методів та засобів, які підтвердили ефктивність запропонованих рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Barford, P.; Kline, J.; Plonka, D.; Ron, A. A Signal Analysis of Network Traffic Anomalies. In Proceedings of the 2nd ACM SigcommWorkshop on Internet Measurment (IMW ’02), Marseille, France, 6–8 November 2002; ACM: New York, NY, USA, 2002; pp. 71–82.
2. Du, Z.; Ma, L.; Li, H.; Li, Q.; Sun, G.; Liu, Z. Network Traffic Anomaly Detection Based on Wavelet Analysis. In Proceedings of the 2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA), Kunming, China, 12–15 June 2018; pp. 94–101.
3. Lu, W.; Ghorbani, A.A. Network Anomaly Detection Based on Wavelet Analysis. EURASIP J. Adv. Signal Process. 2008, 2009, 837601. [CrossRef]
4. Mdini, M.; Blanc, A.; Simon, G.; Barotin, J.; Lecoeuvre, J. Monitoring the network monitoring system: Anomaly Detection using pattern recognition. In Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 8–12 May 2017; pp. 983–986.
5. Chernov, S.; Cochez, M.; Ristaniemi, T. Anomaly Detection Algorithms for the Sleeping Cell Detection in LTE Networks. In Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, UK, 11–14 May 2015; pp. 1–5.
6. Sahinoglu, Z.; Tekinay, S. Multiresolution decomposition and burstiness analysis of traffic traces. In Proceedings of the 1999 IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, LA, USA, 21–24 September 1999; Volume 2, pp. 560–563.