АНОТАЦІЯ

Кваліфікаційна робота складається з пояснювальної записки та графічної частини (ілюстративний матеріал – слайди). Об’єм графічної частини кваліфікаційної роботи становить 10 слайдів. Об’єм пояснювальної записки складає 64 друкованих сторінок формату А4 (210×297), об’єм додатків – 14 друкованих сторінок формату А4. Кваліфікаційна робота складається з шести розділів, в яких нараховується 10 рисунків та 3 таблиці з даними. В роботі використано 20 літературних джерел.

У кваліфікаційній роботі вирішується задача розробки та впровадження розподіленої служби, призначеної для накопичення та аналізування використання ресурсів обчислювальних вузлів, їх активних процесів, виконання розподілених прикладних програм, з метою виявлення відхилень у їх роботі та інформування користувача про нештатні ситуації.

Основні вимоги, які висуваються до цієї служби полягають у здійсненні кластерного моніторингу та аналізу використання ресурсів.

Ключові слова: ВРАЗЛИВОСТІ, АВТОМАТИЗОВАНА СИСТЕМА, СИСТЕМА ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ, РОЗПОДІЛЕНІ КОМП’ЮТЕРНІ СИСТЕМИ

ANNOTATION

Qualification work consists of an explanatory note and a graphic part (illustrative material - slides). The graphic part of the qualifying work is 10 slides. The volume of the explanatory note is 64 printed A4 pages (210 × 297), the volume of appendices is 10 printed A4 pages. The qualification work consists of six sections, in which there are 15 figures and 3 tables with data. 20 literary sources were used in the work.

Qualification work solves the problem of developing and implementing a distributed service designed to accumulate and analyze the use of resources of computer nodes, their active processes, the implementation of distributed applications, to identify deviations in their work and inform the user about abnormal situations.

The main requirements for this service are cluster monitoring and analysis of resource use.

Keywords: VULNERABILITIES, AUTOMATED SYSTEM, VULNERABILITY DETECTION SYSTEM, DISTRIBUTED COMPUTER SYSTEMS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, ОДИНИЦЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД	База Даних
ДД	Доступ до Даних
ІК	Інтерфейс Користувача
ЛП	Логіка програми
ЗПЗ	Зловмисне програмне забезпечення
РОМ	Розподілена Обчислювальна Мережа
ВЗ	Віддалена Загроза
НСД	Несанкціонований Системний Доступ
ПК	Персональний Комп’ютер
ПЗ	Програмне Забезпечення
РПЗ	Руйнуючий Програмний Засіб
ОС	Операційна Система
ВК	Віртуальний Канал
ЕОМ	Електронна обчислювальна машина
CPU	(Central Processing Unit) Центральний Процесор
МБ	Мегабайт
КБ	Кілобайт
LAN	(Local Area Network) Локальна мережа
WAN	(Wide Area Network) Глобальна мережа
API	(Application Programming Interface) Прикладний програмний інтерфейс

ВСТУП

Під час роботи з розподіленими системами, виявлення несправностей у них є складним завданням, оскільки відхилення не обов’язково відразу виявляються за допомогою попереджень або збоїв системи. Особливо це стосується тонких помилок, таких як варіації продуктивності запущеної програми, що не завжди є помилкою програми.

Такі типи проблем не завжди можуть бути виявлені за допомогою звичайних інструментів моніторингу кластерів, оскільки вони розглядають лише показники кластера або розподілені налагоджувачі, відстежуючи тільки певні програми, тому, не знайдуть причини погіршення продуктивності якщо це спричинене іншим джерелом.

Оскільки використання розподілених систем стає все більш поширеним серед тих, хто не має глибоких знань про ці системи, можливість швидкого інформування користувача про будь-які несправності або відхилення від норми сприятиме їх ефективному використанню. Крім того, це допомогло б і розробникам, оскільки вони могли б легко отримати дані про продуктивність своїх програм, не впроваджуючи конкретних процедур для цього завдання і, таким чином, спростити розробку нового розподіленого програмного забезпечення.

Дана робота має на меті з’ясувати, який мінімально необхідний обсяг інформації кожного вузла системи про операційну систему та її процеси потрібний, для виявлення ненормальної роботи. Для цього створюється прототип системи, яка збирає цю інформацію з кластера, і здійснює аналіз даних на предмет можливих несправностей.

Створення системи, здатної збирати великі обсяги даних із кластера, зберігати їх та виконувати елементарний аналіз даних, залишаючи

більшість ресурсів кластерів вільними для своїх обчислень, та створення інструменту моніторингу кластерів з низькими ресурсами, який збиратиме великі обсяги системних даних з високою частотою з кожного з вузлів та аналізуватиме дані є актуальною задачею.

1 АНАЛІЗ АРХІТЕКТУРИ РОЗПОДІЛЕНИХ СИСТЕМ І АРХІТЕКТУРНИХ СТИЛІВ

Одним з основних недоліків розподілених систем є складність базового апаратного та програмного забезпечення. Це розташування зазвичай відоме як топологія. Це те, що забезпечує платформу для розподілених вузлів для спілкування та координації один з одним у разі потреби.

1.1 Розподілена система

Розподілена система — це програмна система, яка з'єднує сукупність гетерогенних незалежних комп'ютерів, де координація та зв'язок між комп'ютерами відбуваються лише за допомогою передачі повідомлень з метою досягнення спільної мети. Ідея розподілених систем полягає в тому, щоб надати зовнішньому світу точку зору єдиної цілісної системи. Отже, сукупність незалежних комп'ютерів або вузлів з'єднані між собою через локальну мережу (LAN) або глобальну мережу (WAN).

1.2 Архітектури розподілених систем

Архітектури розподілених систем об'єднуються з компонентами та роз'ємами. Компоненти можуть бути окремими вузлами або важливими компонентами в архітектурі, тоді як конектори — це ті, що з'єднують кожен із цих компонентів.

Компонент: модульний блок з чітко визначеними інтерфейсами; змінні; багаторазового використання.

З'єднувач: канал зв'язку між модулями, який опосередковує координацію або співпрацю між компонентами.

Отже, ідея розподілених архітектур полягає в тому, щоб ці компоненти були представлені на різних платформах, де компоненти

могли спілкуватися один з одним через комунікаційну мережу для досягнення конкретних цілей.

1.3 Архітектурні стилі

Існує чотири різних архітектурних стилі, а також гібридна архітектура, коли справа доходить до розподілених систем. Основна ідея полягає в тому, щоб організувати логічно різні компоненти та розподілити ці комп’ютери між різними машинами.

1) Багатошарова архітектура

2) Архітектура на основі об'єктів

3) Архітектура, орієнтована на дані

4) Архітектура на основі подій

5) Гібридна архітектура

1.3.1 Багатошарова архітектура

Багатошарова архітектура відокремлює шари компонентів один від одного, надаючи їй набагато більш модульний підхід. Добре відомим прикладом цього є модель OSI, яка включає багатошарову архітектуру під час взаємодії з кожним із компонентів. Кожна взаємодія є послідовною, коли рівень зв’язується з сусіднім шаром, і цей процес триває, доки запит не буде задоволено. Але в деяких випадках реалізацію можна зробити так, що деякі шари будуть пропущені, що називається міжрівневою координацією. Завдяки міжрівневій координації можна отримати кращі результати завдяки підвищенню продуктивності.

1.3.2 Архітектура на основі об'єктів

Цей архітектурний стиль заснований на слабосполученому розміщенні об’єктів. Це не має певної архітектури, як шари. Як і в шарах,

тут немає послідовного набору кроків, які необхідно виконати для даного виклику. Кожен з компонентів називають об’єктами, де кожен об’єкт може взаємодіяти з іншими об’єктами через певний конектор або інтерфейс. Вони набагато більш прямі, коли всі різні компоненти можуть безпосередньо взаємодіяти з іншими компонентами за допомогою прямого виклику методу.

1.3.3 Архітектура, орієнтована на дані

Як видно з назви, ця архітектура заснована на центрі обробки даних, де первинний зв’язок відбувається через центральне сховище даних. Це загальне сховище може бути як активним, так і пасивним. Це більше схоже на проблему споживача виробника. Виробники виробляють товари до загального сховища даних, а споживачі можуть запитувати дані з нього. Це загальне сховище може бути навіть простою базою даних. Але ідея полягає в тому, що спілкування між об’єктами відбувається через це спільне сховище. Це підтримує різні компоненти (або об’єкти), забезпечуючи постійний простір для зберігання цих компонентів (наприклад, базу даних MySQL). Вся інформація, пов’язана з вузлами в системі, зберігається в цьому постійному сховищі. В архітектурах на основі подій дані надсилаються та отримуються лише тими компонентами, які вже підписалися.

Деякі популярні приклади - розподілені файлові системи, виробники-споживачі та веб-сервіси даних.

1.3.4 Архітектура на основі подій

Вся комунікація в такій системі відбувається через події. Коли подія буде згенерована, вона буде відправлена в систему шин. При цьому всі інші будуть сповіщені про те, що така подія сталася. Таким чином, цей вузол може витягнути подію з шини та використовувати її. Іноді ці події

можуть бути даними або навіть URL-адресами ресурсів. Одержувач може отримати доступ до будь-якої інформації, яка надається в події, і обробляти відповідно, процеси спілкуються через поширення подій.

Ці події іноді несуть дані. Перевагою цього архітектурного стилю є те, що компоненти слабо з’єднані. Тому легко додавати, видаляти та змінювати компоненти в системі.

Однією з основних переваг є те, що ці різнорідні компоненти можуть зв'язуватися з шиною за допомогою будь-якого протоколу зв'язку.

1.4 Вимоги до розподілених систем

Щоб досягти основної мети - поліпшення виконання запитів користувача - розподілена система повинна задовольняти деяким необхідним вимогам. Можна сформулювати наступний набір вимог, які повинна задовольняти розподілена обчислювальна система.

1) Відкритість.

Всі протоколи взаємодії компонент всередині розподіленої системи в ідеальному випадку повинні бути засновані на загальнодоступних стандартах. Це дозволяє використовувати для створення компонент різні засоби розробки і операційні системи. Кожна компонента повинна мати точну і повну специфікацію своїх сервісів. В цьому випадку компоненти розподіленої системи можуть бути створені незалежними розробниками. Порушення цієї вимоги унеможливлює створення розподіленої системи, що охоплює кілька незалежних організацій.

2) Масштабованість.

Масштабованість обчислювальних систем має кілька аспектів. Найважливіший з них - можливість додавання в розподілену систему

нових комп'ютерів для збільшення продуктивності системи, що пов'язано з поняттям балансування навантаження (load balancing) на сервери системи. До масштабування ставляться так само питання ефективного розподілу ресурсів сервера, який обслуговує запити клієнтів.

3) Підтримка логічної цілісності даних.

Запит користувача в розподіленій системі повинен або коректно виконуватися цілком, або не виконуватися взагалі. Ситуація, коли частина компонент системи коректно обробили запит, а частина - ні, є найгіршою.

4) Стійкість.

Під стійкістю розуміється можливість дублювання декількома комп'ютерами одних і тих самих функцій або можливість автоматичного розподілу функцій усередині системи в разі виходу з ладу одного з комп'ютерів. В ідеальному випадку це означає повну відсутність унікальної точки збою, тобто вихід з ладу будь-якого комп'ютера не призводить до неможливості обслужити запит користувача.

5) Безпека.

Кожен компонент, який утворює розподілену систему, повинен бути впевнений, що його функції використовуються авторизованими компонентами або користувачами. Дані, що передаються між компонентами, повинні бути захищені як від спотворення, так і від перегляду третіми сторонами.

6) Ефективність.

У вузькому сенсі стосовно до розподілених систем під ефективністю буде розумітися мінімізація накладних витрат, пов'язаних з розподіленим характером системи. Оскільки ефективність у цьому сенсі може

суперечити безпеці, відкритості та надійності системи, слід зазначити, що вимога ефективності в даному контексті є найменш пріоритетним. Наприклад, на підтримку логічної цілісності даних в розподіленій системі можуть витрачатися значні ресурси часу і пам'яті, проте система з недостовірними даними навряд чи потрібна користувачам. Бажаною властивістю проміжного середовища є можливість організації ефективного обміну даними, якщо програмні компоненти, які взаємодіють, знаходяться на одному комп'ютері. Ефективне проміжне середовище повинно мати можливість організації такої взаємодії без порушення стеку TCP/IP. Для цього можуть використовуватися системні сокети (unix sockets) в POSIX системах або іменовані канали (named pipes).

1.5 Проміжне середовище розподілених систем

З точки зору одного з комп'ютерів розподіленої системи, всі інші, що входять до неї є віддаленими обчислювальними системами. Теоретичною основою мережевої взаємодії віддалених систем є загальновідома модель взаємодії відкритих систем OSI/ISO, яка розділяє процес взаємодії двох сторін на сім рівнів: фізичний, канальний, мережевий, транспортний, сеансовий, прикладної, представницький.

У мережах найбільш поширеного стеку протоколів TCP/IP протокол TCP є протоколом транспортного, а протокол IP - протоколом мережевого рівня.

Використання протоколу TCP/IP за допомогою сокетів надає стандартний, міжплатформений, але низькорівневий сервіс для обміну даними між компонентами. Для виконання сформульованих вище вимог до розподілених систем функції сеансового і представницького рівня повинно взяти на себе деяке проміжне середовище (middleware), зване так само проміжним програмним забезпеченням. Таке середовище повинне

допомогти створити розробниками відкриті, масштабовані і стійкі розподілені системи. Для досягнення цієї мети проміжне середовище повинне забезпечити сервіси для взаємодії компонент розподіленої системи. До таких сервісів відносяться:

1) забезпечення єдиного і незалежного від операційної системи механізму використання одними програмними компонентами сервісів інших компонент;

2) забезпечення безпеки розподіленої системи: аутентифікація і авторизація всіх користувачів сервісів компоненти і захист інформації, що передається між компонентами інформації від спотворення і читання третіми сторонами;

3) забезпечення цілісності даних: управління транзакціями, розподіленими між віддаленими компонентами системами;

4) балансування навантаження на сервери з програмними компонентами;

5) виявлення віддалених компонент

В межах однієї розподіленої системи може використовуватися кілька видів проміжних середовищ. При правильному проєктуванні системи кожна її розподілена компонента надає свої сервіси за допомогою єдиного проміжного середовища, і використовує сервіси інших компонент за допомогою єдиного проміжного середовища, однак ці середовища можуть бути різними.

2 АНАЛІЗ ЗАГРОЗ ІНФОРМАЦІЙНІЙ БЕЗПЕЦІ В РОЗПОДІЛЕНИХ СИСТЕМАХ

2.1 Технології розвитку вразливостей і загроз

Вразливості системи безпеки, які не діють у розподіленій системі, можуть бути навмисно використані або ненавмисно запущені. Загрози експлуатації або спрацьовування є лише потенційними і кваліфікуються як напад або нещасний випадок. Ефективне усунення та маскування вразливостей і загроз вимагає аналізу ризиків на основі витрат.

Вразливості існують в апаратному забезпеченні, мережах, операційних системах, системах баз даних і додатках. Щодня відкриваються нові. Інформацію про виявлені вразливості та загрози можна отримати з відомих баз даних інцидентів безпеки або метабаз, таких як ICAT, CERT, vdb або CVE, із систем сповіщення, таких як Cassandra, або з інших джерел інформації про інциденти безпеки.

Багато ідей або алгоритмів з досліджень надійності та відмовостійкості надають корисні аналогії дослідженням безпеки.

Для контролю доступу у відкритих системах потрібна довіра. Існують проблеми з підходами на основі ідентифікації та використанням цифрових облікових даних. Постійні дослідження можуть створити надійні рейтинги довіри для користувача на основі кількох типів доказів, включаючи облікові дані, спостережувану поведінку, рекомендації та репутацію. Рейтинги довіри використовуються для покращення механізму контролю доступу на основі ролей (RBAC).

Довіра та конфіденційність тісно переплетені у взаємодії між суб’єктами, які співпрацюють. Важливо зберегти конфіденційність даних. Об’єкти можуть інкапсулювати політику конфіденційності, уподобання власника та інші метадані разом із даними власника.

Вразливі місця можна визначити за допомогою досліджень шахрайства. Шахрайство можна виявити, виявивши моделі обманної поведінки.

2.2 Модель загроз у розподілених мережах

Вразливість можна визначити як недолік або слабкість у процедурах безпеки системи, дизайні, реалізації або внутрішньому контролі.

Вразливість може бути випадково запущена або навмисно використана, що спричинить порушення безпеки.

Моделювання вразливостей включає аналіз їх особливостей, їх класифікацію та побудову. У літературі доступно багато різноманітних моделей вразливостей у різних середовищах і за різних припущень. Детальний аналіз чотирьох поширених комп'ютерних вразливостей визначає їх характеристики, очікувані політики, порушені їх експлуатацією, а також кроки, необхідні для усунення таких вразливостей у майбутніх випусках програмного забезпечення.

Модель життєвого циклу вразливості була застосована до трьох тематичних досліджень, які показують, як системи залишаються вразливими довго після виправлень безпеки. Протягом свого життя вразливість може перебувати в шести станах: народження, виявлення, розкриття, виправлення, публічність і смерть.

Техніка аналізу на основі моделі для виявлення вразливостей конфігурації в розподілених системах включає формальну специфікацію бажаних властивостей безпеки, абстрактну модель системи, яка фіксує її поведінку, пов'язану з безпекою, і методи перевірки, щоб перевірити, чи задовольняє абстрактна модель властивостям безпеки.

Можна виділити два види вразливостей: операційні та інформаційні. Перші включають неочікуване пошкоджене зв'язок у розподіленій базі

даних, а другі включають несанкціонований доступ (секретність/конфіденційність), несанкціоновану модифікацію (цілісність), аналіз трафіку (проблема висновку) та візантійський вхід.

Вразливості не потрібно вичерпно видаляти, оскільки вони створюють лише потенціал для атаки. Небажано постійно відчувати загрозу вразливості. Вразливості існують не лише через помилки чи упущення, але й можуть бути побічним ефектом законної функції систем. Деякі вразливості існують у системах і не завдають шкоди в її життєвому циклі.

Деякі відомі вразливості доводиться терпіти через економічні чи технологічні обмеження. Видалення інших може зменшити зручність використання. Вимагання пароля не тільки для входу, але й для будь-якого значного запиту ресурсу може зробити його безпечним, але зменшить зручність використання.

Конструкція системи не повинна дозволяти противнику знати про вразливості, невідомі власнику системи.

2.3 Вразливість до шахрайства

Шахрайство можна визначити як обман, навмисно застосовуваний з метою отримання несправедливої або незаконної вигоди. Розкриття конфіденційної інформації неавторизованим особам або несанкціонований продаж списків клієнтів телемаркетологам є шахрайством. Це свідчить про збіг шахрайства з порушенням конфіденційності.

Шахрайство може зробити системи більш вразливими для подальшого шахрайства. Для цього потрібні механізми захисту, щоб уникнути пошкодження в майбутньому.

Шахраїв можна поділити на дві категорії: імітатори та шахраї.

Імітатор — це незаконний користувач, який краде ресурси у жертв, наприклад, заволодіваючи їхніми обліковими записами. Шахраєм, навпаки, є законний користувач, який навмисно отримує вигоду від системи чи інших користувачів шляхом обману. Наприклад, шахраї отримують законні телекомунікаційні рахунки та користуються послугами без наміру оплатити їх.

Шахрайство передбачає зловживання довірою. Шахрай прагне представити себе як надійну людину і друга. Зрозуміло, що чим більше довіряє один іншим, тим вразливішим він стає.

2.4 Проблеми дослідження вразливості

Вразливості, як і помилки, спричиняють збої та атаки. Їх можна охарактеризувати як недоліки в дизайні, реалізації або розгортанні. Серйозність недоліку та його вплив на програму потребує аналізу.

Якісний вплив може бути виражено як низький/середній/високий ступінь деградації з точки зору продуктивності та доступності. Кількісний вплив визначається економічними збитками, вимірними каскадними ефектами та часом, необхідним для відновлення. Це може включати кількісне визначення повторюваності збоїв або атак.

Для ефективного вилучення характеристик і властивостей відомих вразливостей необхідні процедури та методи. Це аналогічно розумінню того, як виникають несправності. Інструменти, які шукають відомі вразливості в метабазах, мають обмеження. Механізми безпеки, які додають або змінюють записи в метабазах, можуть лише низькі, непередбачувані кроки нападника. Характеристики можна дізнатися з поведінки зловмисника.

Моделі життєвого циклу для вразливостей слід будувати після ретельного аналізу вразливостей для даного типу програми або системи, використовуючи їх унікальні характеристики. На кожній фазі життєвого циклу слід визначити кумулятивну вразливість системи та розпізнати найнебезпечніші або найпоширеніші типи вразливостей. Знання ступеня вразливості системи, тривалості фаз життєвого циклу та основних типів вразливостей для даного етапу допоможуть захистити систему від цих типів вразливостей. Найкращі захисні процедури можна адаптивно вибрати із попередньо визначеного набору.

Моделі життєвого циклу повинні допомогти вирішити декілька проблем. По-перше, вони повинні допомогти уникати вразливостей у розгорнутій системі, виявляючи та усуваючи їх на етапах проектування та впровадження. По-друге, вони повинні сприяти оцінці та вимірюванні вразливостей компонентів і підсистем системи та системи в цілому на кожному етапі життєвого циклу. По-третє, моделі допоможуть у найефективнішому виявленні вразливостей у розгорнутій системі до того, як вони почнуть експлуатуватися зловмисником. Вони допомогли б у найбільш ефективному усуненні або маскуванні цих вразливостей, наприклад засновані на принципах, аналогічних відмовостійкості. Крім того, зловмисник може залишатися невідомим або невизначеним щодо важливих параметрів системи, наприклад, через недетерміновану або оманливу поведінку системи чи збільшення різноманітності компонентів або кілька ліній оборони.

Дослідження повинні забезпечити методи оцінки впливу вразливостей на безпеку в програмах і системах. Вони мають створити формальні описи впливу вразливостей та розробити кількісні методи оцінки впливу вразливостей.

Отриманий рейтинг допоможе в аналізі ризиків. Дослідники можуть визначити основні принципи проектування та вказівки для боротьби з вразливими місцями в будь-якій системі.

Також необхідні дослідження щодо вразливостей самих механізмів безпеки та вразливостей через незловмисне використання інформації, яке створює загрози.

2.5 Несанкціонований доступ у розподілених мережах. Механізми його реалізації

Несанкціонований доступ представляє собою спробу порушника отримати доступ до мережних ресурсів без відповідного дозволу. Несанкціонований доступ дозволяє: неавторизовану маніпуляцію даними (читання, модифікацію, копіювання або переміщення файлів, підробку мережних адрес, переключення з'єднань, зміну маршрутів); доступ до системи (реєстрація зі «стороннім» обліковим записом — імітація, маскування, встановлення та розсилка зловмисного програмного забезпечення для здійснення подальших атак, несанкціоноване встановлення й використання мережних з'єднань, несанкціоноване використання комунікаційних протоколів, використання комунікаційних з'єднань для атак, використання хибних налагоджень, використання внутрішніх помилок, відторгнення комунікаційних відношень); підвищення прав доступу (отримання інформації або виконання процедур, що не є доступними при встановленому для користувача рівні доступу).

У розподілених мережах несанкціонований доступ може реалізовуватися такими специфічними прийомами як подолання:

1) систем адміністрування доступом до захищеного інформаційного об'єкта, заснованих на атрибутах користувача (ідентифікатори, паролі, біометричні дані тощо);

2) систем адміністрування доступом до робочих станцій, локальних мереж і т.п., заснованих на атрибутах робочих станцій чи засобів управління доступом і маршрутизації відповідних мереж (файрволів, проксі-серверів, маршрутизаторів тощо).

Останній тип НСД використовує недостатню стійкість відповідних механізмів ідентифікації та автентифікації й може мати характер імітації, маскування довіреного об’єкта або суб’єкта шляхом:

3) підміни довіреного об’єкта або суб’єкта розподіленої обчислювальної системи;

4) впровадження в розподілену обчислювальну систему хибного об’єкта, зокрема, шляхом нав’язування хибного маршруту (зміна маршрутизації) чи з використанням недоліків алгоритмів віддаленого пошуку (атаки типу «людина всередині»).

Імітація має на увазі фальсифікацію (порушення цілісності) IP-адреси, повторне відтворення повідомлень (порушення доступності) з метою захоплення сеансу зв’язку, зміну параметрів маршрутизації й змісту інформації, що передається. Згадана вище недостатня ідентифікація й автентифікація віддалених один від одного об’єктів полягає, перш за все, у труднощах здійснення однозначної ідентифікації повідомлень, переданих між суб’єктами й об’єктами взаємодії. Звичайно, у розподілених обчислювальних системах ця проблема вирішується в такий спосіб: у процесі створення віртуального каналу об’єкти РОМ обмінюються певною інформацією, що унікально ідентифікує даний канал. Такий обмін, звичайно, називається «рукостисканням» (handshake). Для надійної ідентифікації й автентифікації повідомлень, у принципі, можна використати, по-перше, хешфункції, які обчислюються за допомогою, наприклад, відкритого ключа, динамічно виробленого при встановленні

каналу, й, по-друге, випадкові багатобітні лічильники пакетів і мережні адреси станцій. Однак на практиці, наприклад, у протоколі TCP для ідентифікації використаються лише два 32-бітних лічильники.

Відзначимо, що не завжди для зв’язку двох віддалених об’єктів у РОМ створюється віртуальний канал. Практика показує, що найчастіше, особливо для службових повідомлень (наприклад, від маршрутизаторів) використовується передача одиночних повідомлень без підтвердження.

Окрім того відомо, що для адресації повідомлень у розподілених обчислювальних системах використовуються мережні адреси, що є унікальними для кожного об’єкта системи (на канальному рівні моделі OSI — це апаратурна адреса мережного адаптера, на мережному рівні — адреса визначається залежно від використовуваного протоколу мережного рівня (наприклад, IP-адреса). Мережна адреса також може використовуватися для ідентифікації об’єктів розподіленої обчислювальної системи. Однак мережна адреса не є прихованою інформацією й досить просто підробляється. Звідси випливає, що одним із механізмів несанкціонованого доступу є підробка (для об’єктів взаємодії — порушення цілісності) мережних адрес тих об’єктів, що атакують. Тому використовувати мережні адреси, як єдиний засіб ідентифікації об’єктів, неприпустимо.

У цьому випадку є можливою також типова віддалена атака, яка полягає в передачі мережею повідомлень від імені довільного об’єкта або суб’єкта РОМ (маскування).

Реалізація механізму віддалених атак щодо несанкціонованого доступу з підробкою (порушенням цілісності) мережних адрес звичайно складається з:

1) передачі пакетів обміну з атакуючого об’єкта на мету атаки від імені довіреного суб’єкта взаємодії (при цьому передані повідомлення будуть сприйняті системою як коректні);

2) передачі службових повідомлень від імені мережних керуючих пристроїв, наприклад, від імені маршрутизаторів.

Наступним механізмом несанкціонованого доступу є зміна параметрів маршрутизації. Це пов’язано з тією особливістю РОМ, що сучасні глобальні мережі представляють собою сукупність сегментів мережі, пов’язаних між собою через мережні вузли. Для забезпечення ефективної й оптимальної маршрутизації в розподілених обчислювальних системах застосовуються спеціальні керуючі протоколи, що дозволяють маршрутизаторам обмінюватись інформацією один з одним, повідомляти хости про новий маршрут, віддалено управляти маршрутизаторами.

При цьому абсолютно очевидно, що маршрутизація в глобальних мережах відіграє найважливішу роль і, як наслідок цього, може піддаватися атаці. Основна мета атаки, пов’язаної з нав’язуванням хибного маршруту, полягає в тому, щоб змінити (порушити цілісність, модифікувати) вихідну маршрутизацію на об’єкті розподіленої обчислювальної системи так, щоб новий маршрут проходив через хибний об’єкт — хост атакуючого. Реалізація даної типової віддаленої атаки складається в несанкціонованому використанні протоколів керування мережею для зміни вихідних таблиць маршрутизації.

Для зміни маршрутизації атакуючому необхідно послати по мережі для протоколів керування мережею спеціальні службові повідомлення від імені мережних керуючих пристроїв (наприклад, маршрутизаторів). У результаті успішної зміни маршруту атакуючий одержить повний контроль (можливість порушення конфіденційності, цілісності та

доступності) над потоком інформації, яким обмінюються два об’єкти розподіленої обчислювальної системи, і атака перейде до другої стадії, пов’язаної з прийманням, аналізом і передачею повідомлень, одержуваних, наприклад, від дезінформованих (у разі порушення цілісності) об’єктів РОМ.

У розподіленій обчислювальній системі часто виявляється, що її віддалені об’єкти не мають досить інформації, що необхідна для адресації повідомлень. Звичайно, такою інформацією є апаратурні (адреса мережного адаптера) і логічні (наприклад, IP-адреса) адреси об’єктів РОМ. Для одержання подібної інформації в розподілених обчислювальних системах використовуються різні алгоритми віддаленого пошуку, що полягають у передачі мережею спеціального виду пошукових запитів і очікуванні відповідей на запит із шуканою інформацією.

Після одержання відповіді на запит суб’єкт РОМ, що запросив, має всі необхідні дані для адресації. Керуючись отриманими з відповіді відомостями про шуканий об’єкт, суб’єкт РОМ, що запросив, починає адресуватися до нього, маючи можливості порушення конфіденційності та доступності певної інформації чи шуканого об’єкта.

У випадку використання розподіленої обчислювальної системи механізмів віддаленого пошуку існує можливість на атакуючому об’єкті перехопити надісланий запит і надіслати на нього хибну відповідь (здійснити маскування), де вказати дані, використання яких приведе до адресації на атакуючий хибний об’єкт. Надалі весь потік інформації між суб’єктом й об’єктом взаємодії буде проходити через хибний об’єкт РОМ.

Ще одним механізмом несанкціонованого доступу є використання недоліків алгоритму віддаленого пошуку, наслідком чого є впровадження зловмисниками в систему об’єктів, що мають назву «Хибних об’єктів

РОМ». Цей варіант впровадження в РОМ хибного об’єкта складається в періодичній передачі на об’єкт, що атакується, заздалегідь підготовленої хибної відповіді без приймання пошукового запиту. Справді, для того щоб послати хибну відповідь, не завжди обов’язково чекати приймання запиту (може, в принципі, не бути можливості перехоплення запиту). При цьому атакуючий може спровокувати об’єкт, що атакується, на передачу пошукового запиту, і тоді його хибна відповідь буде негайно мати успіх. Дана типова віддалена атака надзвичайно характерна для глобальних мереж, коли просто немає можливості перехопити пошуковий запит. Використання хибного об’єкта для організації віддаленої атаки на розподілену обчислювальну систему дає змогу одержати контроль над потоком інформації між об’єктами й застосовувати різні методи впливу на перехоплену інформацію. Одним із таких методів є селекція потоку інформації й збереження її на хибному об’єкті РОМ шляхом перехоплення переданої між суб’єктом й об’єктом взаємодії інформації.

3 ДОСЛІДЖЕННЯ МЕХАНІЗМІВ ЗМЕНШЕННЯ ВРАЗЛИВОСТІ ТА ЗАГРОЗ

3.1 Безпека інформації

Інформація, яка зберігається в базі даних, дуже важлива для компанії. Для будь-якої організації важливо зберігати дані для майбутніх цілей і отримувати їх або маніпулювати ними для подальшого використання. Існують три основні обмеження, які необхідно підтримувати щодо інформації:

1) Цілісність – цілісність інформації означає, що інформація має бути повною і точний, його не слід змінювати без дозволу його законного користувача.

2) Конфіденційність - конфіденційність інформації означає, що інформація не повинна бути доступною або використаною неавторизованими користувачами.

3) Доступність – доступність інформації означає, що будь-яка інформація, яка потрібна законному користувачеві/системі в будь-який час, має бути доступною для цього користувача/системи.

Коли приватна мережа будь-якої компанії підключена до Інтернету або розподіленої системи, збільшуються шанси атаки на внутрішню мережу, інформація та багато додатків, які використовують цю інформацію, можуть працювати на різних комп’ютерах. Для безпеки на цих машинах або пристроях слід встановити систему виявлення вторгнення.

Дизайн пристрою зберігання також впливає на послуги розподіленої системи. Дизайн пристрою зберігання даних повинен мати такі функції, що дані повинні бути швидко доступні клієнту в різних географічних

місцях, а особисті й приватні деталі повинні контролюватися. Зараз для підтримки безпеки даних у розподіленій системі використовуються дві основні методики: реплікація та обмін секретами.

3.2 Фізична безпека в розподіленій системі

Усі елементи розподіленої системи мають бути фізично захищені. Деякі процедури внутрішнього контролю необхідні для всього апаратного та програмного забезпечення, розгорнутого в розподілених і менш безпечних середовищах.

Рівень безпеки, що оточує будь-яке обладнання та програмне забезпечення, повинен залежати від чутливості даних, до яких можна отримати доступ, важливості оброблених додатків, вартості обладнання та наявності резервного обладнання. Через свою портативність і розташування в розподіленому середовищі персональні комп’ютери часто є основною мішенню для крадіжок і неправомірного використання. Розташування ПК та чутливість даних і систем, до яких вони мають доступ, визначають ступінь необхідної фізичної безпеки. У таких випадках деяким компаніям, установам слід розглянути можливість закріплення комп’ютерів на робочих станціях, блокування або видалення дисків і непотрібних фізичних портів, а також використання паролів або автоматичних тайм-аутів. Співробітники також повинні мати лише доступ до ПК та даних, необхідних їм для виконання своєї роботи. Ефективність заходів безпеки залежить від проінформованості співробітників та забезпечення виконання цих заходів контролю. Фізична безпека мереж, а також ПК включає захист живлення, фізичні замки та безпечні робочі зони, забезпечені охоронцями. Фізичний доступ до компонентів мережі (файлів, програм, комунікацій тощо) повинен бути обмежений тими, кому потрібен доступ для виконання своїх завдань. ПК або мережеві робочі станції повинні бути захищені паролем і відстежуватися на предмет

активності робочої станції. Усі ці дії впливають на послуги, що надаються розподіленою системою.

3.3 Безпека мережі та політика аутентифікації

Технічна безпека мережі в розподіленій системі є основним напрямком безпеки розподіленого середовища. В основному це стосується того, як ми захищаємо мережу від атак, пов'язаних із мережею, і як ми працюємо з механізмом аутентифікації.

3.3.1 Брандмауер

Брандмауер – це система, яка є єдиною точкою з'єднання внутрішньої мережі та захищає її від зовнішньої мережі. По суті, брандмауер захищає мережу від несанкціонованого трафіку, відфільтровуючи небажаний трафік, що надходить у захищену мережу або виходить із неї. Існують певні правила прийняття рішень у технології брандмауера на основі цих правил брандмауер фільтрує пакет даних. Ці правила засновані на попередньо визначених політиках безпеки. Маршрутизатори також є корисною точкою перекриття для всього трафіку, що входить або виходить з мережі. У деяких випадках зловмисник може приховати фактичну адресу пакета даних і зробити таку адресу, як пакет, належним до внутрішньої мережі призначення, надіслати до внутрішньої мережі, тобто пакети, які стверджують, що надходять з внутрішніх машин, але насправді надходять з за межами, оскільки такі пакети зазвичай є частиною адресних атак. У таких атаках зловмисник прикидається, що походить із внутрішньої машини. Тому в таких випадках приймати рішення такого роду можна лише за допомогою фільтруючого маршрутизатора на периметрі вашої мережі. Тільки фільтруючий маршрутизатор у тому місці, яке є кордоном між «внутрішньою» та «зовнішньою» мережею, здатний розпізнати такий пакет, подивившись на

адресу джерела та чи надійшов пакет із внутрішнього мережевого з’єднання чи із зовнішнього мережевого з’єднання.

3.3.2 Протокол Kerberos

У зв’язку з зростаючим використанням Інтернету та технологій для ведення бізнесу, організація може використовувати техніку шифрування для захисту конфіденційної інформації, що передається через Інтернет та інші мережі. в області онлайн-додатків і обчислень клієнтського сервера, де зв'язок здійснюється на основі TCP/IP, використовується протокол Kerberos. Kerberos — це система аутентифікації, яка використовується в розподіленій системі. Його беруть на озброєння багато підприємств, організацій, університетів. Kerberos використовує концепцію криптографії. Він надає докази особистості принципа для захисту від атак, пов’язаних із ідентифікацією. У роботі Kerberos принципал є головною дійовою особою, а принципал, як правило, є або користувачем, або певною службою на домашній машині. Принципал складається з трьох кортежів: <основне ім'я, екземпляр, область>. Якщо принципал є користувачем — справжньою особою — первинне ім’я — це ідентифікатор входу, який може бути будь-яким ідентифікатором електронної пошти або ім’ям користувача, який є унікальним для кожного користувача, а екземпляр або нульовий, або представляє певні атрибути користувача, який є root. Якщо принципал не є користувачем, а є службою системи, тоді ім’я служби використовується як основне ім’я, а ім’я машини – як екземпляр, тобто rlogin.myhost. Область використовується для розрізнення різних доменів аутентифікації.

3.4 Розробка механізмів захисту

У розподілену систему мають бути вбудовані механізми захисту, за допомогою яких можна реалізувати різні правила захисту, при цьому

варто враховувати три важливих ознаки: фокус керування, багаторівневу організацію механізмів захисту і простоту.

3.4.1. Фокус керування

Організувати захист прикладних програм (зокрема розподілених) можна, використовуючи три основних підходи. Перший підхід – це захист, безпосередньо асоційований з прикладною програмою, пов’язаною з доступом та обробкою даних, тобто незалежно від операцій, які можуть здійснюватися нею з елементами даних, має бути збережена цілісність даних. Зазвичай такий захист використовують в системах баз даних, тоді заздалегідь визначають різні обмеження цілісності, які потім автоматично перевіряють у процесі кожної модифікації елемента даних.

Другий підхід – це захист, який застосовує точну вказівку щодо того, хто і як може використати операції доступу до даних або ресурсів. У цьому разі фокус керування тісно пов'язаний з механізмами контролю доступу. Цей підхід також використовується з метою деталізації керування доступом.

Третій підхід – зосередити увагу безпосередньо на користувачеві, вживаючи заходів, щоб доступ до прикладного програмного забезпечення або його ресурсів мали тільки визначені люди, незалежно від операцій, які вони збираються виконати.

3.4.2. Багаторівнева організація механізмів захисту

Під час розробки систем захисту важливо визначитись, скільки рівнів повинні мати механізми захисту. Рівень у цьому контексті відповідає логічній організації системи.

Комбінуючи багаторівневі структури інформаційно-телекомунікаційних мереж і розподілених систем, отримуємо схему, подану на Рис. 1.

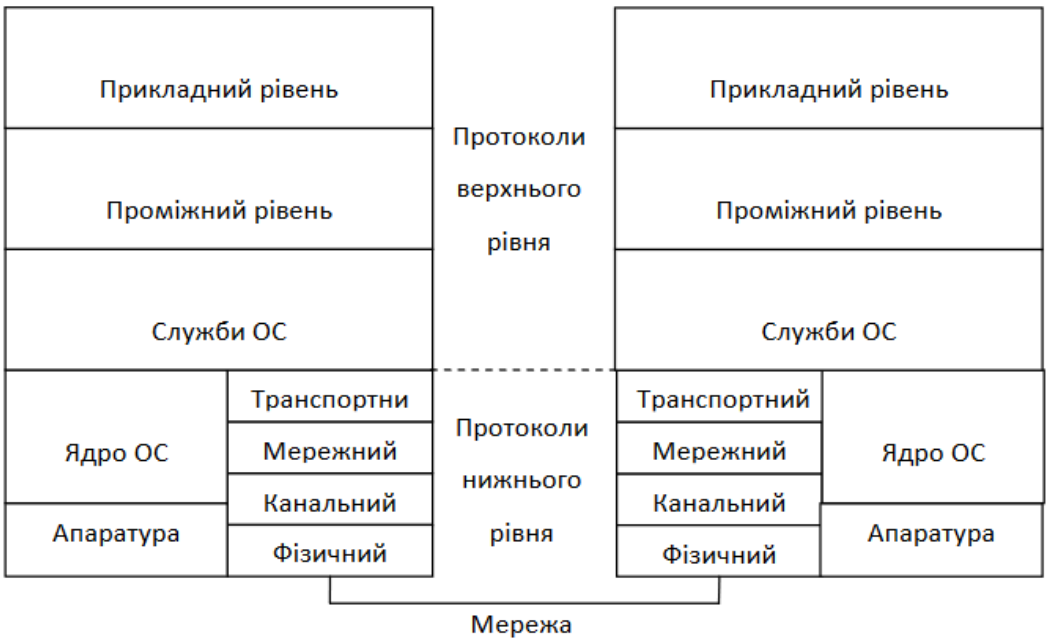

Рисунок 1 Логічна багаторівнева організація розподілених систем

На Рис. 1 служби загального призначення відділено від комунікаційних служб. Цей поділ дуже важливий для розуміння розподілу за рівнями механізмів захисту розподілених систем і, зокрема, для уявлення про довіру, причому різниця між поняттями «довіра» і «захист» є суттєвою. Система може бути або не бути захищеною, але думка клієнта про те, що система захищена – це питання довіри. Рівень, на якому розміщують механізм захисту, залежить від довіри клієнта до захисту його служб.

3.4.3. Розподіл механізмів захисту

Залежності між службами, що вимагають довіри, призводять до виникнення поняття «довірена обчислювальна база» (Trusted Computing Base, TSB), яка є набором усіх механізмів захисту розподіленої системи,

необхідних для реалізації правил захисту, при цьому чим менший TSB, тим краще. TSB у розподіленій системі може містити локальні операційні системи різних хостів. Якщо розподілена система побудована на основі проміжного рівня (надбудови над наявною мережною операційною системою), її захист залежатиме від захисту базових локальних операційних систем.

Розподілені системи, побудовані з використанням технологій проміжного рівня, вимагають довіри до локальної операційної системи, під керуванням якої вони працюють. Якщо такої довіри немає, то частина функціональності локальної операційної системи має бути вбудована в саму розподілену систему. Враховуючи, що в операційній системі є мікроядро, більшість служб виконані у вигляді звичайних процесів користувача, то, наприклад, стандартна файлова система може бути повністю замінена іншою, особливо пристосованою до специфічних потреб розподіленої системи, зокрема до різних механізмів захисту. У такому разі доцільно відділити служби захисту від інших видів служб за рахунок розподілу цих служб по різних машинах відповідно до того ступеня захисту, який їм необхідний.

3.5 Захищені канали

Побудова захищеної розподіленої системи передбачає дві основні складові: побудову захищеного зв'язку між клієнтом і сервером та спосіб авторизації.

Захищений зв'язок вимагає аутентифікації взаємодіючих сторін, що гарантує цілісність повідомлень, а можливо, й їх конфіденційність.

Спосіб авторизації визначає яким чином серверу, який одержав запит від клієнта, розпізнати факт авторизації клієнта для передачі цього

запиту на обробку. Авторизація потрібна, щоб здійснювати керований доступ до ресурсів.

Захист зв'язку між клієнтами і серверами є організацією між з’єднаними сторонами захищеного каналу (secure channel). Захищені канали оберігають відправника й одержувача від перехоплення, модифікації та підроблення повідомлення. Зазвичай немає потреби вводити захист від переривання зв'язку, бо це не призведе до отримання повідомлення зловмисником. Захист повідомлень від перехоплення здійснюється за рахунок гарантованої конфіденційності: захищені канали гарантують, що повідомлення в них не можуть перехопити зловмисники. Захист повідомлень від модифікації або підроблення забезпечується за допомогою протоколів взаємної аутентифікації та цілісності повідомлень.

3.6 Контроль доступу

У моделі клієнт-сервер після встановлення між клієнтом і сервером захищеного каналу клієнт створював запити, які передавалися серверу та містили операції над ресурсами, контрольованими сервером. Як правило, сервер об'єктів керує декількома об'єктами. Запит клієнта зазвичай містить звертання до методу конкретного об'єкта та може бути виконаний лише за умови, що клієнт має достатні для цього права доступу (access rights).

Формальне підтвердження прав доступу називають контролем доступу (access control), у той час як видачу прав доступу - авторизацією (authorization). Ці два поняття тісно взаємопов'язані, і нерідко одне з них використовують замість другого.

Існує дуже багато методів контролю доступу. Один з особливо значущих методів контролю доступу до ресурсів – це створення брандмауера, що захищає прикладне програмне забезпечення або навіть усю мережу цілком.

4 НАУКОВО-ДОСЛІДНА ЧАСТИНА

4.1 Дослідження існуючих способів побудови

Кластери комп’ютерів, що утворюють розподілену систему, широко використовуються для створення паралельних обчислювальних систем, це високомасштабований та економічно ефективний спосіб створення такої системи. З огляду на те, що ці великі системи стають все більш складними для користувачів, виникає необхідність відстежувати справність такої системи.

Поширеним підходом для цього є наявність системи моніторингу кластера, такого як **ganglia** або **HP Insight Cluster Management Utility**, які відповідають за збір даних, таких як інформація про загальний центральний процесор (CPU) і використання пам'яті для кожного вузла кластера та представлення його користувачеві. Це полегшує порівняння загального використання кожного з вузлів кластера. Однак є проблеми з цими системами; вони зазвичай працюють із повільним тактом для оновлення (отже, можливо, що короткострокові системні відхилення залишаються непоміченими), інша проблема полягає в тому, що вони зазвичай не дають жодних даних про процеси. Однак деякі системи дозволяють користувачеві додавати додаткові точки даних для моніторингу, **ganglia** надає можливість самостійно визначати користувачеві метрики, які можуть представляти довільний стан, але тут кожна метрична характеристика повинна бути чітко визначена. Але тоді потрібно заздалегідь визначитися з необхідними характеристиками, які стануть цінними.

Існують системи, які відстежують конкретні програми, з або без, включення метрики хост-системи. Вони відрізняються в залежності від того, чи запускаються і зупиняються одночасно з процесом, який вони

відстежують, чи сервер працює безперервно. Зазвичай вони зберігають лише зібрані дані та мають інтерфейс для користувача, щоб отримувати поточні або історичні дані з бази даних.

Системи з сервером, який працює безперервно, можуть мати агентів, які збирають системні метрики, на додаток до отримання даних із контрольованої програми через встановлену бібліотеку, яка дозволяє їй підключатися та передавати власні показники використання на сервер.

Хоча для системного адміністратора життєво важливо переконатися, що система в цілому працює належним чином, це не обов'язково є основною проблемою того, хто використовує систему, зазвичай вони піклуються лише про один процес або невелику групу процесів, залежно від використання ними системи. Для перевірки справності окремої розподіленої програми виникає така ж проблема, як і при перевірці системи в цілому, це займає дуже багато часу, якщо вона не автоматизована якимось чином.

Наприклад, можна перевірити кожен вузол окремо, щоб переконатися, що програма працює належним чином, або отримати інформацію з інструментів, що використовуються для запуску програми в кластері, таких як **ansible**. Перевірка кожного вузла на наявність помилок або ненормальної роботи займає багато часу і може бути важкою навіть для користувачів, знайомих із системою. Хоча використання таких інструментів, як ansible, дає лише інформацію про виконання через певні проміжки часу та лише інформацію про те, чи працює програма, а не про її правильність.

Однак слід знати те, які саме вимірювання є нормальними для добре налаштованої системи, де все працює належним чином. Це, звичайно, буде відрізнятися залежно від того, що робить розподілена система в певний

момент часу. Велике використання CPU, коли система не працює над обчисленнями, або відсутність використання під час обчислень інтенсивного CPU, могло б бути ознакою відхилень, оскільки це не очікуваний показник для випадку використання. Те саме стосується кожного аспекту відстежуваних показників, велике використання пам’яті в деяких програмах, які зазвичай використовують лише кілька мегабайт (МБ), може свідчити про ненормальну поведінку. Тому, щоб виявити дивну поведінку в системі, необхідно знати, чим показники відрізняються від звичайної поведінки, це, звичайно, важко дізнатися.

Якби відстежувалися лише системні показники, як, наприклад, у загальних інструментах моніторингу кластерів, це було б дуже складним питанням, але якщо ми також відстежуємо показники кожного з процесів, ми все одно знаємо, як повинен поводитися кожен окремий процес, навіть якщо вони працюють одночасно, хоча ми не обов’язково знаємо, як поводитиметься система в цілому. Ця відсутність знань є меншою проблемою, якщо знати, як повинні поводитися всі компоненти системи, оскільки все ще можна виявити відхилення у кожному з цих процесів.

4.2 Кластерний моніторинг

У цьому розділі дається вступ до кластерного моніторингу та представлені існуючі системи моніторингу кластера, перш ніж розглядати способи аналізу великої кількості зібраних даних; для виявлення будь-яких відхилень від очікуваних показників.

4.2.1 Загальна характеристика моніторингу кластерів

Традиційні системи моніторингу кластерів, як правило, дотримуються архітектури клієнт-сервер, показаної на рисунку 2, де на кожному вузлі є певна форма агента, яка отримує деяку системну інформацію. Потім інформація надсилається на серверну частину для

моніторингу. Далі сервер збирає, зберігає та візуалізує дані кластера для користувача. Дані можна збирати з ОС, особливо в системі типу Unix, де їх можна легко збирати з файлової системи **proc (procfs)**, також можна отримати цю інформацію з систем та/або програм середнього програмного забезпечення. Ці дані збираються через певні проміжки часу, які зазвичай тривають кілька секунд, щоб утримувати накладні витрати настільки низькими, наскільки це можливо, щоб вони не заважали нічому, що працює на кластері.

Збираються тільки загальні системні дані, тобто середнє використання ресурсів процесора за останні 5, 10 і 15 хвилин та використання пам'яті.

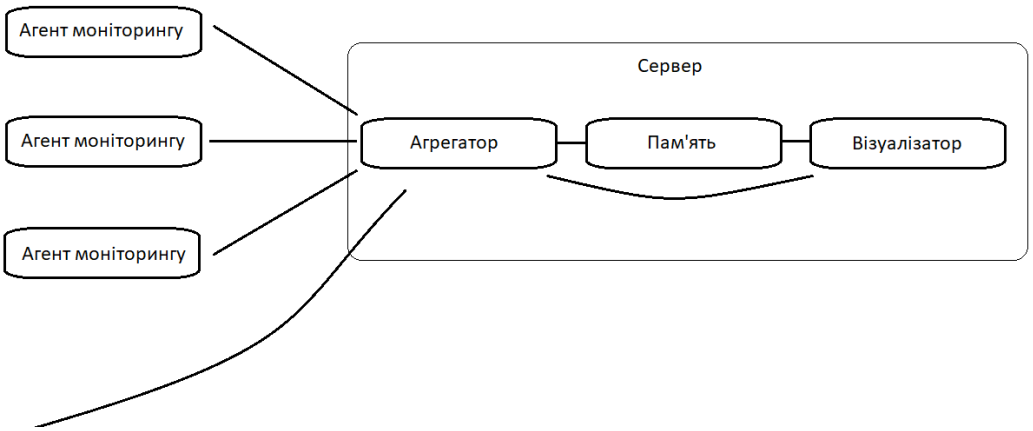

Рисунок 2 Загальна архітектура традиційних систем моніторингу кластерів

Зазвичай обробка даних на вузлах не відбувається, агенти лише збирають інформацію. Така поведінка пояснюється необхідністю зменшення впливу агентів на розподілену систему. Також існує заздалегідь визначена фіксована конфігурація, встановлена на початку, для того, щоб дані надсилалися. Для великих кластерів система моніторингу налаштована як деревоподібна структура, де кожен агент надсилає інформацію агрегатам, які потім надсилають дані у більш компактній. Якщо виникає помилка, її можна вирішити шляхом

надсилання даних з будь-якої точки структури в декілька місць, тим самим отримуючи надмірність у системі.

Ganglia використовує подібну архітектуру, що складається з двох частин, які працюють на кожному вузлі кластера і не надсилають зібрані дані на вузол сервера, замість цього вони використовують багатоадресну розсилку та надсилають їх кожному вузлу кластера, щоб кожен вузол знав стан всього кластера. Тоді сервер може опитувати дані з будь-якого одного вузла, таким чином, опитуючи кілька вузлів, вони досягають надмірності. Це також може бути використано для кластерів, розташованих у глобальній мережі, де вони створюють ієрархічне дерево моніторингу за допомогою кількох своїх агрегованих частин сервера.

Тенденція розвитку цих типів систем зосереджена переважно на масштабованості. Розподілені системи швидко збільшуються у розмірах, тому їх моніторинг та збереження у працездатному стані стає більш складнішим завданням. Отже, збереження зручності використання та можливість використовувати одну і ту ж систему, незалежно від того, чи складається кластер з 5 або 5000 вузлів є перевагою.

Існують системи, націлені на моніторинг додатків. Однак вони відрізняються від системного моніторингу тим, що швидкість збору має бути значно вищою, що призводить до того, що обсяг зібраних даних буде значно збільшений. Загальною рисою систем моніторингу для конкретних програм є те, що вони призначені для моніторингу конкретних завдань, отже, час виконання збігається з роботою, за якою здійснюється моніторинг, і більша частина аналізу проводиться пізніше.

4.2.2 Способи автоматизування виявлення несправностей

Існує спосіб автоматизувати процес виявлення несправностей у розподіленій системі шляхом використання комбінації статистики та контрольованого виявлення.

Автоматизація цього процесу є логічним кроком, оскільки збільшення розміру моніторингових систем, частоти збирання та збільшення кількості зібраних даних для аналізу стає великою проблемою. Тому це неможливо зробити одній особі, особливо для аналізу в реальному часі.

Можна поєднати використання аналізу сповіщень про некоректну роботу та інцидентів з алгоритмом навчання на основі правил з критеріями складності для створення набору правил прогнозування. Потенційні умови моніторингу потім будуються на основі цих правил, у разі будь-якого погіршення життєво важливих показників системи (визначених прийнятними порогами або умовами моніторингу), проблема позначається та надсилається допоміжному персоналу. Вирішення цієї проблеми потім використовується для оновлення умов системи. Таким чином створюється система, яка стає точнішою, щодо якої проблеми з часом позначаються та надсилаються допоміжному персоналу.

4.2.3 Альтернативні структури моніторингу

Багатоадресне спілкування **ganglia** між агентами моніторингу в межах кластера надає кожному агенту показники для всього кластера, щоб отримати всі показники шляхом опитування одного агента. Натомість розроблена система передає дані кожного агента моніторингу безпосередньо на сервер. Інша відмінність - це частина сервера, де **ganglia** візуалізує дані користувачеві. Натомість система яка розробляється в ході

кваліфікаційної роботи проводить аналіз даних та попереджає користувача у разі виявлення певних відхилень.

Можна розглянути альтернативну структуру моніторингу яка використовує колектори, які називаються кінцевими хостами, які збирають показники та надсилають повідомлення зворотного зв'язку координатору, який потім пересилає дані на сервер зворотного зв'язку. Платформа моніторингу періодично зв'язується з цими серверами зворотного зв'язку для отримання доступних агрегованих даних вимірювань, для аналізу та візуалізації в режимі реального часу. Обидві системи використовують протокол передачі гіпертекстового протоколу (HTTP) для будь-якої комунікації.

Astrolabe використовує протокол однорангового зв’язку з ієрархічною структурою зон. Зона — це або хост, або набір зон, що не перекриваються, створюючи таким чином деревоподібну структуру. Кожен вузол має агента, який збирає інформацію, а також виступає в якості веб-сервера. Результати створюються за допомогою SQL-запитів. Таким чином можна збирати, поширювати та узагальнювати інформацію про зони.

Panopticon, як і **Astrolabe**, використовує ієрархію агентів, які працюють на кожному вузлі, у деревоподібній структурі. Ці агенти спостерігають і записують метрики хост-системи та надають їх іншим вузлам за запитом. Таким чином інформація поширюється вгору по дереву, записуючи маршрут у даних. Кореневий вузол отримує всі дані, при цьому йому доводиться спілкуватися лише з кількома дочірніми вузлами. Під час проведення експериментів **Panopticon** вони використовували подібний невеликий набір показників, як **ganglia**. Між агентами використовується бінарний протокол клієнт-сервер.

5 СПЕЦІАЛЬНА ЧАСТИНА

5.1 Архітектура і дизайн розробленої системи

Система розробляється для роботи під ОС Linux. Це розподілена система моніторингу, що складається з агентів-колекторів, що працюють на кожному вузлі кластера, вони надсилають повідомлення на сервер механізму аналізу, який обробляє інформацію (див. Рис. 3).

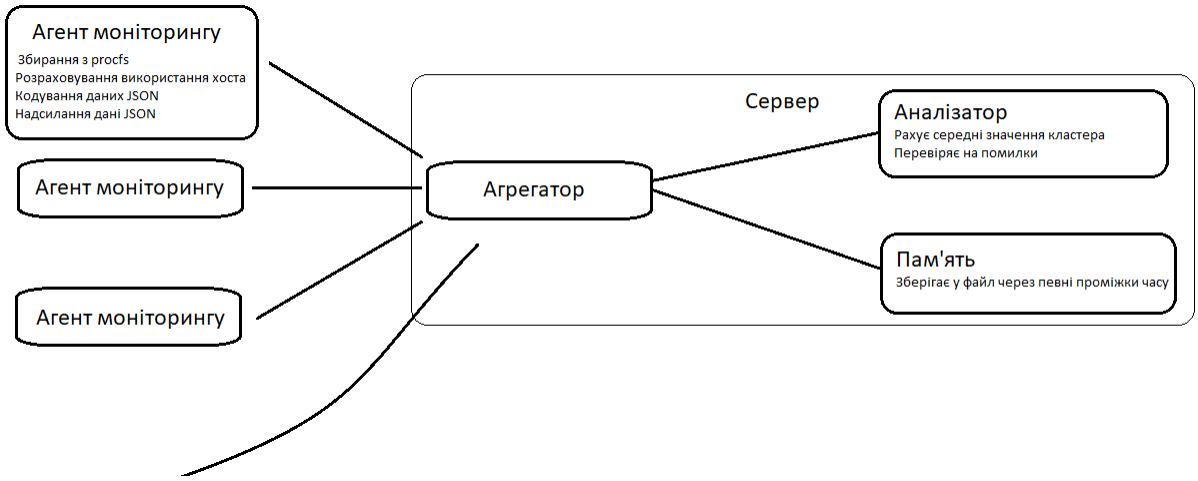

Рисунок 3 Архітектура розподіленої системи моніторингу

5.1.1 Принцип роботи колектора

Колектор системи - це компонент, який працює на кожному вузлі кластера. Колектор працює на таймері інтервалів, на початку кожного з цих інтервалів він збирає метрики про ОС і кожен процес, що виконується ОС. Усі зібрані дані потім перетворюються в json перед відправкою в механізм аналізу методом push. Потім колектор чекає початку наступного інтервалу таймера.

5.1.1.1 Збирання метрик процесів

Усі дані, які збираються, отримані від **procfs**. Метрики ОС збираються з */proc/stat*, */proc/net/dev* і */proc/meminfo*, а для кожного

процесу (PID), що виконується в даний момент, вони збираються з */procPID/stat, /proc/PID/statm* та */proc/PID/io*.

Оскільки невеликі обчислення виконуються на основі даних, надсилаються не всі поля ЦП. Натомість, для кожного запису ЦП у */proc/stat* (це буде весь ЦП плюс кожен потік ЦП, див. табл. 1) поля додаються, щоб зробити загальну кількість галок, і на основі цього обчислюється відсоток використання з моменту останнього збору. Таким чином, для кожного поля ЦП надсилається відсоток використання.

cpu	25220	1	3887	3316908	2450	0	5	0	0 0
cpu0	5057	0	1250	410265	1319	0	1	0	0 0
cpu1	2236	0	258	415867	315	0	1	0	0 0
cpu2	6150	0	938	411482	252	0	2	0	0 0
cpu3	916	0	161	417502	184	0	1	0	0 0
cpu4	3962	0	516	413836	220	0	0	0	0 0
cpu5	1193	0	152	417260	43	0	0	0	0 0
cpu6	4982	0	422	413052	69	0	0	0	0 0

Табл. 1 Дані з файлу */proc/stat*

З файлу */proc/net/dev* збираються 1-е та 8-е числові поля, див. табл. 2, це загальна кількість отриманих та надісланих байтів відповідно. Ці значення зберігаються в колекторі й використовуються для обчислення загального мережевого трафіку в кілобайтах (КБ) з часу останнього збору даних. Цей загальний мережевий трафік надсилається до системи аналізу.

eth0:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0													
lo:	43412	460	0	0	0	0	0	0	43412	460	0	0	0		
0	0	0													

Табл. 2 Приклад уривка файлу */proc/net/dev*

Зібрані показники пам’яті – це поля MemTotal, MemFree, Buffers і Cached з файлу */proc/meminfo*, див. табл. 3. Поле MemTotal – це загальна пам’ять, доступна системі, три інші поля – це вільна пам’ять. Використовуючи ці показники, обчислюється відсоток використання пам’яті, і лише ця обчислена метрика надсилається в механізм аналізу.

MemTotal:	16348432	kB
MemFree:	13830324	kB
MemAvailablee:	14623324	kB
Buffers:	69932	kB
Cached:	1155036	kB
SwapCached:	0	kB
Active:	1490204	kB
Inactive:	752820	kB

Табл. 3 Приклад уривка файлу */proc/meminfo*

5.1.1.2 Передача даних

Дана розробка використовує передачу даних на основі push для надсилання зібраних метрик на сервер.

5.1.2 Опис роботи механізму аналізу

Аналізатор виступає як системний сервер, оскільки саме він об’єднує, зберігає та аналізує дані, отримані від клієнтів. Будь-які з’єднання з цим сервером здійснюються через HTTP, а далі через інтерфейс програмного програмування (API) із перенесенням репрезентативних станів (ResT), з метричними даними, які кодуються як json.

5.1.2.1 Інтерфейс

Колектори використовують ResTful HTTP API для надсилання даних до механізму аналізу. Це включає ім’я хоста, який його надіслав (див. Рис. 4).

```
POST /raw/HOSTNAME HTTP/1.1
Content-Length: X

MESSAGE-BODY
```

Рисунок 4 HTTP - заголовок для передачі даних

5.1.2.2 Агрегування та зберігання

Отримані дані json спочатку аналізуються, потім вони фільтруються та зберігаються у структурі пам'яті, використовуючи ієрархічне формування, розділене на вузли та процеси (див. Рис. 5).

Вся ця структура через регулярні проміжки часу та після завершення роботи зберігається у файлі на диску. Регулярне збереження структури на диску гарантує, що не всі дані будуть втрачені, наприклад, якщо сервер вийде з ладу, і, отже, не вдасться записати вміст на диск.

Фільтрація проводиться для кожної точки даних, отримані дані порівнюються з останнім записаним значенням з того самого вузла. Щойно отримані дані зберігаються лише в тому випадку, якщо немає попередніх даних, або якщо вони відрізняються від попередньої точки даних, усі дані зберігаються з позначкою часу, таким чином можна побачити, як довго значення залишалося незмінним.

Відмітка часу, яка використовується при зберіганні даних, генерується сервером, тому вона відповідає часу отримання даних, а не часу їх збирання, це запобігає помилкам сервера, викликаним різним часом на вузлах кластера.

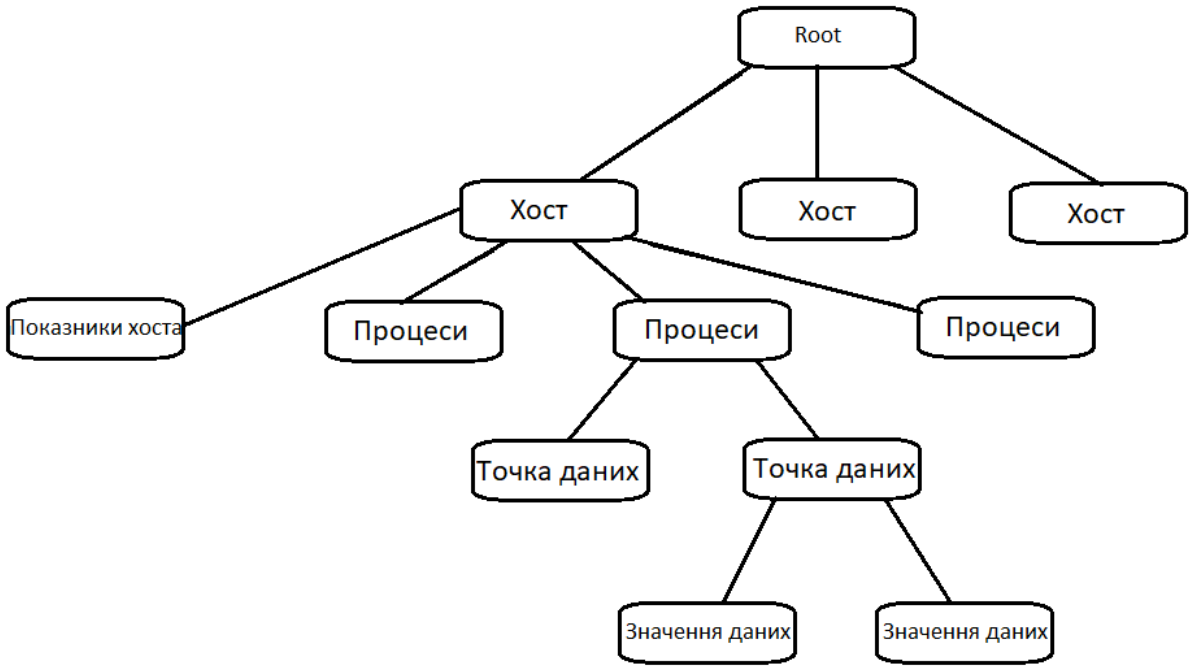

Рисунок 5 Структура пам'яті

5.1.2.3 Аналіз зібраних даних

При аналізі даних перше, що потрібно зробити, це обчислити середні показники кластера для кожного вузла, використовуючи цю інформацію разом з показниками кожного вузла, визначається, чи мають якісь вузли значення, що відрізняються більше, ніж певна сума від середнього. Перелік вузлів, які потрапляють у цю категорію, надходять користувачеві.

Для кожного процесу перевіряється стан, щоб побачити, чи хтось став неактивним.

Використання ресурсів процесора також перевіряється та реєструється, якщо воно перевищує певну суму.

5.2 Реалізація системи

Розроблена система моніторингу реалізована для роботи під Linux.

Кожен вузол тестового кластера запускає 64-розрядну версію Ubuntu 14.04 LTS з ядром Linux 3.13.

На основі ідеї було визначено архітектуру, а потім розроблено та впроваджено кожну частину системи, далі проведені експерименти, щоб перевірити її працездатність та роботу належним чином.

Щоразу, коли виявлялася проблема, тестована програма не працювала належним чином або з дуже поганою продуктивністю, процес повторювали, покращуючи дизайн та реалізацію (див. Рис. 6).

5.2.1 Запобігання помилок при роботі з колектором

При зборі даних з **procfs** завжди є ймовірність, що дозвіл користувача на програму встановлено неправильно, особливо це стосується файлу вводу/виводу процесу (io) для системних процесів. Щоб запобігти помилці кожного разу, коли відбувається збір даних, вони зберігаються і відкривається файл з ними. Це запобігає переповненню журналу помилок однаковими повідомленнями, щоб можна було легше виявити більш серйозні помилоки. Ця проблема з дозволами найчастіше виникає з системними процесами.

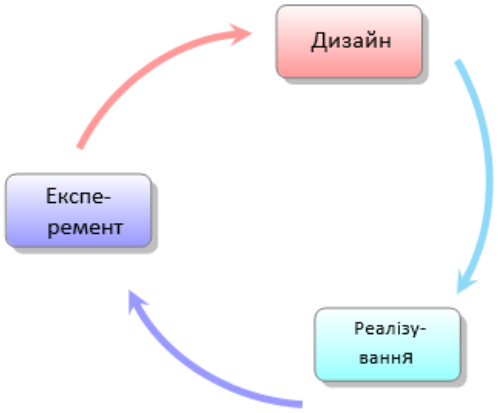

Рисунок 6 Процедура впровадження

Щоб зменшити кількість даних, що надсилаються кожного разу, повні дані для кожного процесу надсилаються лише один раз і позначаються як сталі.

5.2.2 Передача даних для аналізу та зберігання

5.2.2.1 Передача зібраних даних

Для передачі даних перша спроба була зроблена за допомогою двійкового коду, а не json. Повідомлення, закодовані в двійковому коді, були набагато меншими за розміром, ніж еквівалент json, однак перевага в розмірі була втрачена, коли помилка спричинила пошкодження даних, а так як вони були двійковими, їх було дуже важко відновити. Використання json, а не двійкового коду спростила відновлення даних, і помилку було швидко виправлено.

Для серверного аналізу json використовувалася стороння бібліотека під назвою JSMN. Ця бібліотека виявилася дуже зручною, як за розміром кодової бази, так і за дуже низьким споживанням ресурсів.

5.2.2.2 Зберігання даних

Оскільки бази даних, як правило, оптимізовані для читання, було вирішено зберегти всі дані, необхідні для аналізу, в пам’яті.

5.3 Експерементальна частина

5.3.1 Опис експериментальної установки

5.3.1.1 Платформа

Експерименти проводилися на панелі Display Wall Cluster Tromsø. Цей кластер складається з 29 вузлів, підключених через повнодуплексний гігабітний Ethernet. Устаткування кожного вузла складається з:

1) Чотириядерний процесор Intel Xeon W3550 з тактовою частотою 3,07 ГГц із гіперпотоковістю (всього 8 потоків).

2) 12 ГБ оперативної пам'яті (ОЗУ).

3) Відеокарта MSI GeForce GTX 560Ti з 1 ГБ відеопам’яті GDDR5 і PhysX PCI-Express 2.0.

Кластер організовано таким чином, що один з вузлів функціонує як віддалена точка входу для кластера.

5.3.2 Використання пам’яті колектора

Використання ЦП колекторів визначається на основі необроблених даних, які вони зібрали під час кількох виконання.

Відсоток використання ЦП був знайдений шляхом ділення кількості тактів ЦП, використаних кожним колектором, на загальну кількість тактів ЦП від їхніх відповідних хостів протягом того ж періоду часу та множення цього значення на сто.

Використання пам’яті визначається з поля vsize з файлу */proc/PID/stat*, це поле містить загальне використання пам’яті процесом у байтах.

5.3.2.1 Мережа

Використання мережі визначається шляхом збору кількості байтів, надісланих від колектора щоразу, коли він передає дані. Потім ці дані використовуються для визначення середнього розміру повідомлення.

5.3.3 Хід аналізу

5.3.3.1 Порівняльний аналіз

Функція годинника з бібліотеки мови програмування C «time.h» була використана для порівняння того, як ресурси були витрачені в аналізі.

Віднімаючи час початку від часу завершення для кожного виклику функції або сегмента коду, можна дізнатися кількість часу, використаного (у тактах ЦП), для кожної з цих функцій або сегментів.

Використовуючи ці дані, виявляється, який сегмент коду використовує найбільший відсоток часу в порівнянні із загальним часом для кожного циклу.

5.3.3.2 Використання механізму аналізу

Аналіз для сервера здійснюється так само, як і для колекторів, використовуючи вихідні дані, зібрані колектором на тому самому вузлі кластера.

На відміну від колекторів, очікується, що використання ресурсів для аналізу буде зростати зі збільшенням кількості підключених колекторів, а також із збільшенням тривалості виконання. Таким чином, вимірювання виконуються для кількох процесів, з різною кількістю підключених колекторів протягом обмеженого часу (5 хвилин).

5.3.3.3 Зберігання даних

Загальне використання пам’яті за певний час обчислюється шляхом додавання розмірів файлів для кожного часу зберігання.

5.3.5 Експеременти з виявленням вразливостей

Щоб перевірити виявлення відхилень в роботі, на тестовій машині було запущено кілька колекторів і аналізатор.

5.4 Отримані результати

5.4.1 Вимірювання колектора

Результати кожного виконання показали, що використання колектора залишалося практично незмінним з часом, тому для метрики колектора було розраховано середнє значення.

Використання ЦП становило в середньому 0,785 відсотка, це було стабільно в колекторах, жоден не показував вище 1 відсотка використання ЦП. Середнє використання пам’яті під час цих експериментів становило 2,8 МБ.

Очікується стабільне використання, оскільки робоче навантаження та необхідне виділення пам’яті залишаються незмінними, вони можуть змінитися лише в разі зміни кількості процесів на хості.

5.4.1.1 Мережевий трафік

Середній розмір повідомлення від колектора до сервера становить 126,2 КБ, розмір залишається стабільним, як і очікувалося, тому що, як і у випадку використання пам’яті, розмір повідомлення змінюється лише в разі зміни кількості процесів.

Мережевий трафік для всього кластера становитиме 13,175 гігабайт (ГБ) на годину.

5.4.2 Результати продуктивності аналізу

5.4.2.1 Контрольні показники

Порівняльний аналіз коду для системи аналізу показав, що близько 99 відсотків часу було витрачено на два сегменти, ресурси між ними було розподілено досить рівномірно.

Першою з них була функція синтаксичного аналізу JSMN json, було виявлено, що для великих файлів це стосується лише увімкненого PARENT_LINKS. Час, після ввімкнення цієї функції скоротився приблизно на 93 відсотки.

Іншим трудомістким сегментом був код, який зберігає проаналізовані дані json у структурі пам’яті. Для цього завдання використовувався рекурсивний алгоритм, якій згодом був заміненим нерекурсивним, після часові рамки скоротилися приблизно на 75 відсотків.

Об’єднані зусилля цих двох оптимізацій зменшили загальне використання сервера приблизно на 82 відсотки.

5.4.2.2 Показники центального процесора

Використання пам’яті протягом 2-хвилинного часу виконання, для різної кількості вузлів, використання показує майже лінійне збільшення використання на основі кількості підключених вузлів, див. Рис. 7.

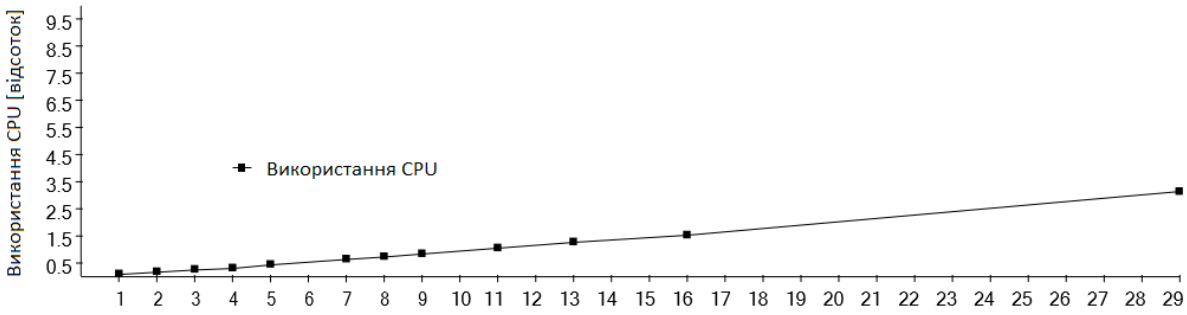

Рисунок 7 Використання ЦП з різною кількістю вузлів.

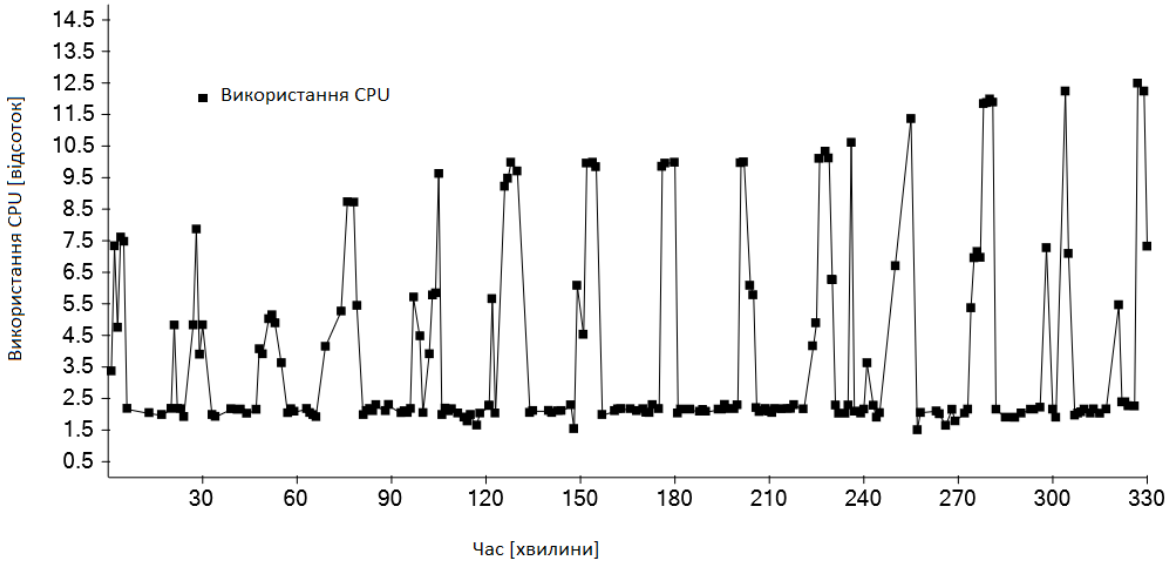

Рисунок 8 Використання ЦП, довгострокове з 29 підключеними вузлами

5.4.2.3 Зберігання

Використання пам’яті протягом тривалого часу для всього кластера, це включає як пам’ять, так і дисковий простір, що використовується, див. Рис. 9.

Це показує, що існує лінійне збільшення використання пам’яті з часом, при цьому використання диска збільшується в геометричній прогресії. Це пов’язано з тим, що повна структура пам’яті щоразу зберігається на диску, тим самим збільшуючи використовуваний дисковий простір із розміром використання пам’яті під час кожного збереження.

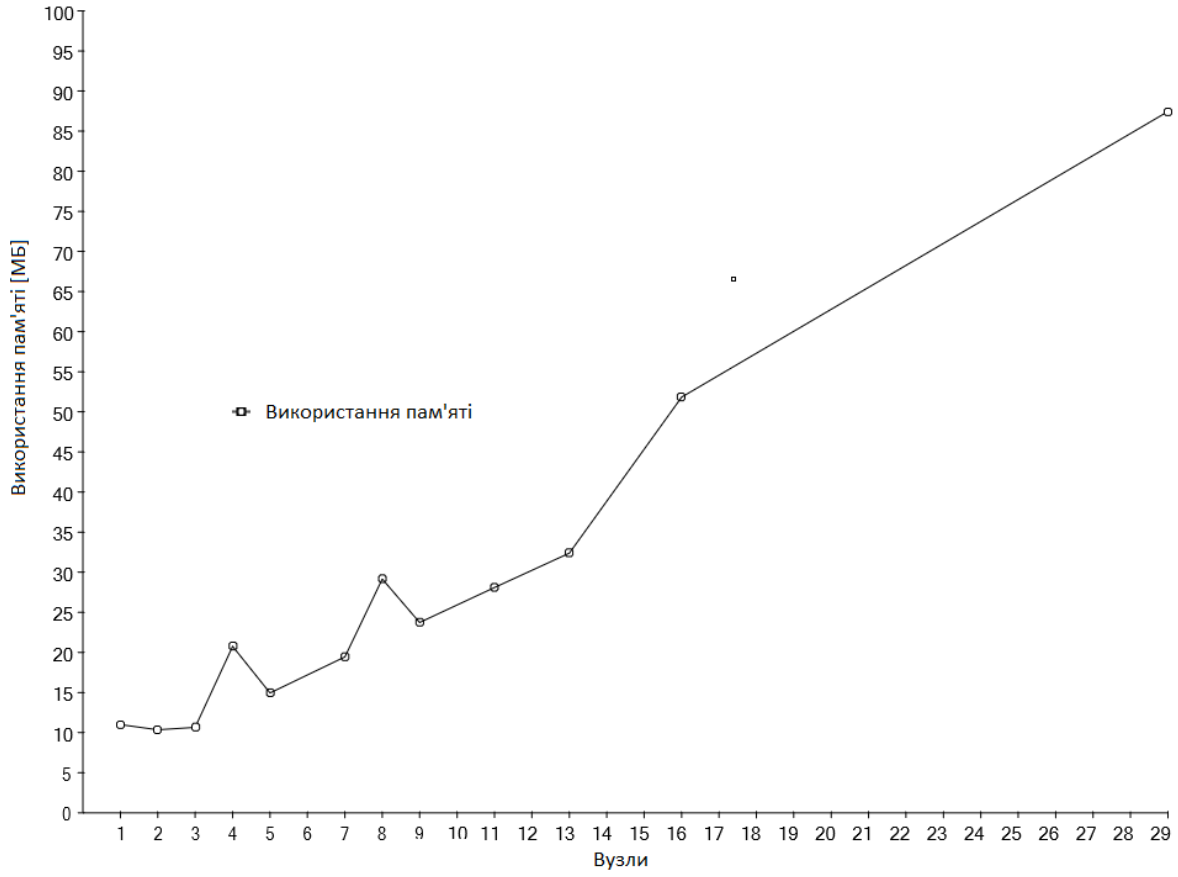

Рисунок 9 Використання пам’яті з різною кількістю вузлів

5.4.3 Аналіз даних

Аналіз даних показав, що поля даних для кожного процесу, які найчастіше змінювалися, були пов’язані з:

1) Процесором

2) Пам’яттю

3) Вводом-виводом

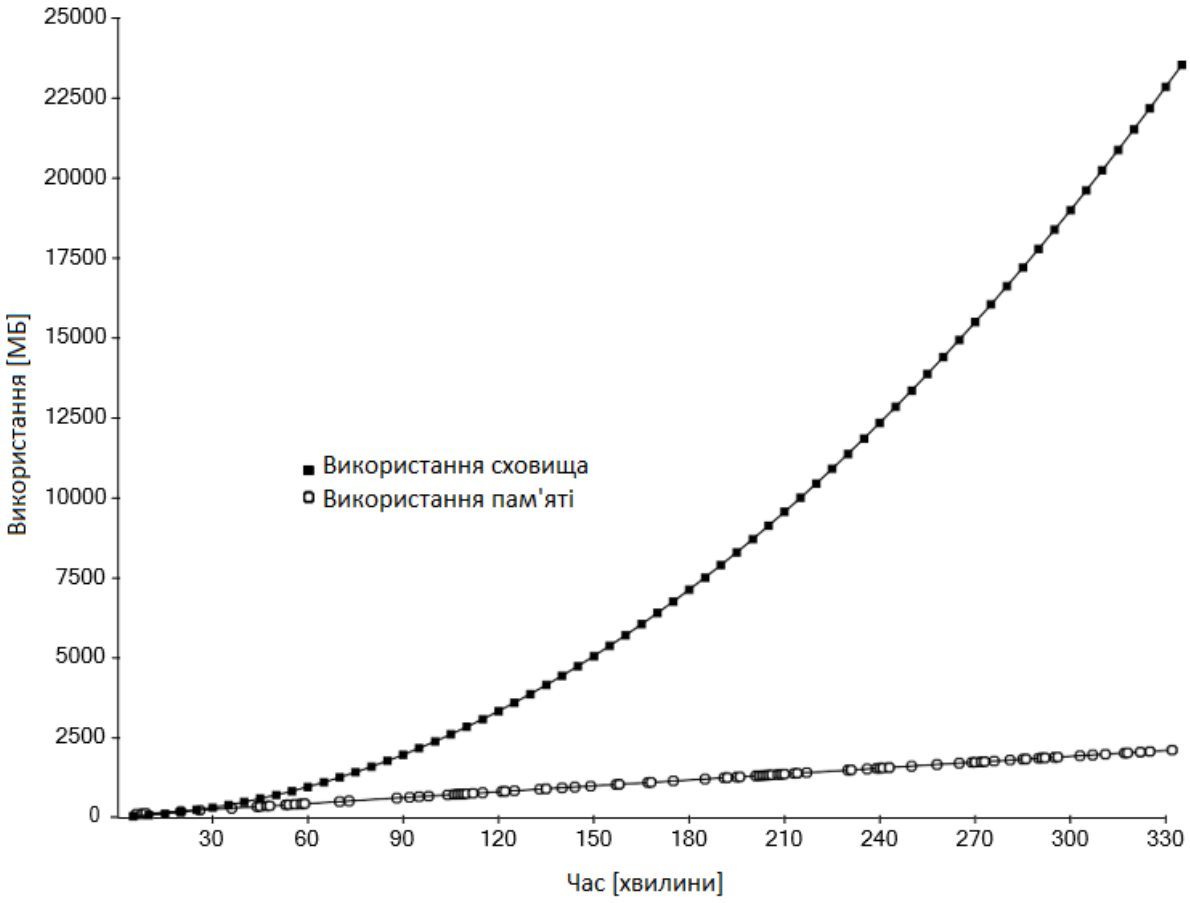

Рисунок 10 Використання сховища, довгострокове з 29 підключеними вузлами.

Поля, пов'язані з аргументом командного рядка, початковими значеннями тощо, ніколи не змінювалися. Це очікувано, оскільки вони показують лише початкові стани процесу, і вони не можуть змінитися пізніше.

5.4.4 Результати виявлення відхилень

За допомогою тесту для виявлення відхилень, було отримано результат про введені значення ЦП і пам’яті. Вони виявилися відхилені від норми кластера, як і очікувалося.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1 Безпека виконання робіт

Безпека виконання робіт включає застосування раціональних методів технології та організацію виробництва. Зокрема, велику роль відіграє зміст праці, форми побудови трудових процесів, ступінь спеціалізації працівників, вибір режимів праці та відпочинку, дисципліна праці, психологічний клімат у колективі, організація санітарного та побутового забезпечення праці.

У формуванні безпечних умов праці також велике значення має враховування медичних протипоказань до використання працівників в окремих технологічних процесах, а також навчання й інструктажі з безпечних методів проведення робіт.

До осіб, які допущені до участі у виробничому процесі, ставляться вимоги щодо відповідності їх фізичних, психофізичних і, в окремих випадках, антропометричних даних характеру роботи. Перевірка стану здоров'я працівників має проводитися як перед допуском їх до роботи, так і періодично у процесі роботи згідно з чинними нормативами. Періодичність контролю стану їх здоров'я визначається залежно від небезпечних та шкідливих факторів виробничого процесу у порядку, встановленому Міністерством охорони здоров'я.

Особи, які допускаються до участі у виробничому процесі, повинні мати відповідну професійну підготовку (у тому числі і з безпеки праці), що відповідає характеру робіт. Навчання працівників щодо охорони праці проводять на усіх підприємствах і в організаціях незалежно від характеру та ступеня небезпеки виробництва відповідно до Типового положення про порядок проведення навчання та перевірки знань з питань охорони праці.

6.2 Значення автоматизації виробничих процесів в питаннях охорони праці

Основними напрямами щодо підвищення рівня безпеки праці має бути комплексна механізація й автоматизація виробництва, що є передумовою докорінного покращення умов праці, зростання продуктивності праці та якості продукції, що, в свою чергу, сприятиме ліквідації відмінності між розумовою і фізичною працею. В той же час, при автоматизації виробничих процесів необхідно враховувати психічні та

фізіологічні фактори, тобто узгоджувати функції автоматичних пристроїв з діяльністю людини-оператора, зокрема, необхідно враховувати антропометричні дані останньої та її можливості до сприйняття інформації (ДСТУ ISO 14738:2013; ГОСТ 21829-76).

У автоматизованому виробництві необхідне також суворе виконання вимог безпеки під час ремонту й налагодження автоматичних машин та систем. Одним із перспективних напрямів комплексної автоматизації виробничих процесів є використання промислових роботів. У цьому випадку між людиною та машиною (технологічним обладнанням) з'являється проміжна ланка – промисловий робот, і система набуває такої структури: людина-промисловий робот-машина. У цьому випадку людина виводиться із сфери постійного безпосереднього контакту з виробничим обладнанням. Ергономічне проєктування та оцінка якості центрів керування та автоматизованих робочих місць повинно здійснюватися відповідно до вимог ДСТУ ISO 11064-6:2013, ДСТУ 8605:2015 та ДСТУ 8603:2015. Забезпечення комфортних та безпечних умов праці на робочих місцях операторів повинно здійснюватися згідно з вимогами ДСТУ 7299:2013.

6.3 Долікарська допомога при пораненнях

При будь-якому порушенні цілісності шкіри і глибоко розташованих тканин необхідно обробити шкіру навколо рани розчином йоду, спиртом тощо. Не рекомендується промивати рану водою або дезінфікуючим розчином. Після обробки рани необхідно накласти асептичну пов'язку. Пов'язка захищає рану від забруднення, інфікування, зменшує біль, а вигляд перев'язаної рани заспокоює хворого.

Обробка рани потребує додержання таких правил: перед обробкою рани необхідно помити руки (якщо поряд немає води, їх слід протерти спиртом або бензином):

— невеликі поранення, садна після обробки шкіри навколо них настойкою йоду або перекисом водню заклеюють лейкопластиром чи медичним клеєм БФ-6;

— не можна видаляти із ран сторонні тіла або бруд, тому, що можна пошкодити судини і викликати кровотечу;

— шкіру навколо рани протирають від країв до периферії шматочком марлі, бинта або вати, яка змочена спиртом, спиртовим розчином йоду чи бензином. (не можна заливати рану йодом!)

— із бинта або індивідуального пакета зробити салфетку такого розміру, щоб вона закривала усю рану, накласти її на ранову поверхню, забинтувати або приклеїти смужками лейкопластиру;

— якщо в рані видно внутрішні органи, мозок або сухожилля, потрібно акуратно накласти стерильну пов'язку, щоб у рану не потрапила інфекція, або краще накрити рану стерильним матеріалом.

При проникаючих пораненнях перша допомога спрямована на запобігання інфікування рани. Проникаючі поранення грудної клітки можуть супроводжуватись кровохарканням, проникненням повітря до підшкірної клітковини, яке виглядає як набряк, але викликає хруст при прощупуванні, накопиченням повітря у плевральній порожнині (пневмоторакс), накопиченням крові у плевральній порожнині (гемоторакс).

Пневмоторакс - це скупчення атмосферного повітря в плевральній порожнині, яке потрапляє через відкриту рану грудної клітки (відкритий пневмоторакс), або при ушкодженні легені чи бронха (закритий пневмоторакс). При його виникненні можливе балотування органів середостіння, що супроводжується розладом кровообігу і дихання. Перша допомога при проникаючих пораненнях грудної клітки має бути спрямована на ліквідацію пневмотораксу, попередження шоку, захист рани від інфікування. Необхідно надати потерпілому напівсидяче положення; накласти герметичну пов'язку, щоб зробити перепону для попадання повітря до плевральної порожнини. Для цього після обробки рани, її закривають смужками лейкопластиру, який накладається у вигляді черепиці. Можна використовувати обгортку від індивідуального перев'язочного пакету, клейонку, целофановий пакет, серветки, що оброблені вазеліном, які потім фіксуються бинтовою пов'язкою до грудної клітки.

Перша допомога при проникаючих пораненнях органів черевної порожнини. Частіше за все це вогнепальні і колото-різані рани. Для всіх поранень черевної порожнини характерним є різкий біль у животі, напруження м`язів черевної стінки (живіт, як "дошка") і симптоми внутрішньої кровотечі, шоку й колапсу. Значна кровотеча спостерігається при ушкодженнях паренхіматозних органів (печінка, селезінка, нирки). Розвиваються характерні симптоми кровотечі: наростає слабкість, настає загальна блідість, нудота, блювання, похолодіння кінцівок. Пульс частий і слабкий, артеріальний тиск падає. Поранення органів шлунково-кишкового тракту супроводжується виходженням у вільну черевну

порожнину кишкового вмісту та інфікуванням її, внаслідок чого швидко розвивається запалення очеревини (перитоніт).

У всіх випадках проникаючих поранень черевної порожнини необхідно обробити рану і накласти асептичну пов'язку. Петлі кишечнику і сальник, які випали в рану, у черевну порожнину не вправляти. Якщо не має абсолютних симптомів проникаючого поранення у живіт, знеболюючих засобів не призначати. Прийом води і їжі категорично забороняється. Пораненого у живіт необхідно негайно госпіталізувати, транспортувати його у положенні лежачи на носилках. При проникаючих пораненнях черевної порожнини показана екстрена операція в перші години, поки не розвинулась інфекція і хворий не втратив багато крові.

ВИСНОВКИ

Проведений аналіз найбільш поширених загрози інформаційній безпеці в розподілених системах, а також їхніх способів реалізації. У ході аналізу виявилося, що базовими принципами інформаційної безпеки є забезпечення цілісності інформації, її конфеденційності і водночас доступності для всіх авторизованих користувачів.

Досліджено засоби захисту, які дають змогу суттєво підвищити рівень захищеності інформації в розподілених системах. Розглянуто можливі ситуації, які виникають та вимагають захисту, а саме перехоплення, переривання, модифікація, підроблення. Було детально досліджено розробки механізмів захисту, їхні особливості та ефективність.

У даній кваліфікаційній роботі була розроблена автоматизована система виявлення вразливостей у розподілених комп’ютерних системах. Дана система покликана виявляти відхилення у роботі центрального процесора, використанні пам’яті для кожного вузла, мережевого трафіку. Розроблена система – це розподілена система моніторингу, що складається з агентів-колекторів, що працюють на кожному вузлі кластера, вони надсилають повідомлення на сервер механізму аналізу, який обробляє інформацію.

Створена розробка була протестована і отримані результати свідчать про те, що розроблена система є ефективною та стійкою для використання, так як проблем не було виявлено.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Буров Є.В., Митник М.М., Комп'ютерні мережі. Том 1. / Є.В. Буров, М.М. Митник // Навчальний посібник – Львів, «Магнолія 2006», 2019. 256 с.

2. Буров Є.В., Митник М.М., Комп'ютерні мережі. Том 2. / Є.В. Буров, М.М. Митник // Навчальний посібник – Львів, «Магнолія 2006», 2019. 334 с.

3. Закон України «Про внесення змін до законів України щодо інформаційної безпеки» веб-сайт. URL: http://search.ligazakon.ua/l_doc2.nsf/link1/JH77G00A.html

4. Закон України «Про основні засади забезпечення кібербезпеки України» веб-сайт. URL: https://zakon.rada.gov.ua/laws/show/2163-19

5. Митник М.М., Микитишин А.Г., Стухляк П.Д., Комплексна безпека інформаційних мережевих систем / М.М. Митник, А.Г. Микитишин, П.Д. Стухляк // Навчальний посібник, 2016. 261 с.

6. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Комп'ютерні мережі. Книга 1. / А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник // Навчальний посібник - Львів, «Магнолія 2006», 2013. 256 с.

7. Микитишин А.Г., Митник М.М., Стухляк П.Д., Пасічник В.В. Комп'ютерні мережі. Книга 2. / А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник // Навчальний посібник – Львів, «Магнолія 2006», 2014. 312 с.

8. Качинський А.Б. Безпека складних систем / Качинський А.Б. -К.: ТОВ «Видавництво «Юстон», 2017. 498 с.

9. Грайворонський М.В. Сучасні підходи до забезпечення кібернетичної безпеки / М.В. Грайворонський // Матеріали XVII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики», НТУУ «КПІ», 2015 р.

10. Anderson J.M. Why we need a new definition of information security / J.M. Anderson // Computers & Security, 2003. 308 с.

11. Understanding difference between Cyber Security & Information Security – CISO Platform, 2016. веб-сайт. URL: http://www.cisoplatform.com/profiles/blogs/understanding-difference-between-cyber-security-information

12. Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network. веб-сайт. URL: http://redwave.net/books/hackg/index/html

13. On the Security of Today's Online Electronic Banking Systems веб-сайт. URL: http://docseurope.electrocomponents.com/b8156853c.pdf

14. Хорев П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев // Учеб. Пособие для студ. Высш.учеб. заведений – М.: Издательский центр – Академия, 2005. 8 с.

15. Остапов С.Е. Технології захисту інформації / С.Е. Остапов, С.П. Євсеєв, О.Г. Король // навчальний посібник – Х.: ХНЕУ, 2013. 476 с.

16. Ляшенко І.О. Європейські критерії безпеки інформаційних технологій / І.О. Ляшенко // Сучасні інформаційні технології у сфері безпеки та оборони, 2012. 84 с.

17. Василюк В. Об’єкти захисту інформації. Методи та засоби захисту інформації / В. Василюк // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, 2006. 88 с.

18. Перелік засобів технічного захисту інформації, дозволених для забезпечення технічного захисту державних інформаційних ресурсів та інформації веб-сайт. URL: https://www.dsszzi.gov.ua/dstszi/control/uk/publish/article?art_id=234237&cat_id=39181

19. Developments in the field of information and telecommunications in the context of international security веб-сайт. URL: www.un.org/en/ga/search/view_doc.asp?symbol=A/RES/64/25

20. Шелухин О.И. Обнаружение вторжений в компьютерные сети (сетевые аномалии) / О.И. Шелухин, Д.Ж. Сакалема, А.С. Филинова. – М.: Горячая линия – Телеком, 2013. 221 с.