Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-сервісу для моніторингу роботи серверного обладнання

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Суховерша В. О.
(підпис) (прізвище та ініціали)

Керівник _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Крамар О.І.
(підпис) (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Суховерша Володимир Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-сервісу для моніторингу роботи серверного обладнання

Керівник роботи Матійчук Любомир Павлович, к.е.н, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від 02 » березня 2021 року № 4/7-171

2. Термін подання студентом завершеної роботи 23 червня 2021р.

3. Вихідні дані до роботи Літературні джерела за тематикою дослідження

4. Зміст роботи (перелік питань, які потрібно розробити)

Вступ. Розділ 1. Аналіз предметної області та проектування WEB-сервісу моніторингу серверного обладнання 1.1. Задачі моніторингу серверного обладнання та порівняльний аналіз відомих програмно-апаратних засобів моніторингу серверного обладнання. 1.2. Аналіз відомих програмних технологій моніторингу серверного обладнання специфікація вимог до системи. 1.3. Розроблення архітектури програмної системи проектування структури бази даних. Розділ 2. Програмна реалізація WEB-сервісу моніторингу серверного обладнання, тестування та дослідна експлуатація. 2.1. Встановлення та базове налаштування системи Nagios.2.2. Тестування. 2.3. Інструкція користувача. 3. Безпека життєдіяльності, основи хорони праці. Висновки. Перелік джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

Основні завдання. Схема віддаленого управління сервером. Програмно-апаратні засоби моніторингу серверного обладнання. Аналіз відомих програмних технологій моніторингу серверного обладнання. Аналіз відомих програмних технологій моніторингу серверного обладнання. Структура системи моніторингу. Діаграма станів процесу моніторингу. Архітектура реалізації. Архітектура web-сервісу. Алгоритм роботи компонента-агента. Програмна реалізація. Ієрархія класів модульного тестування проекту. Інструкція користувача. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 7 червня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.06.2021	*Виконано*
2.	Підбір джерел предметної області та проектування	08.06.2021-09.06.2021	*Виконано*
3.	Переклад та опрацювання джерел про WEB-сервіси моніторингу серверного обладнання …	10.06.2021-11.06.2021	*Виконано*
4.	Виконання дослідження щодо розробки веб-сервісу для моніторингу роботи серверного обладнання	12.06.2021-13.06.2021	*Виконано*
5.	Оформлення розділу «Аналіз предметної області та проектування WEB-сервісу моніторингу серверного обладнання»	14.06.2021-15.06.2021	*Виконано*
6.	Оформлення розділу «Програмна реалізація WEB-сервісу моніторингу серверного обладнання, тестування та дослідна експлуатація»	16.06.2021-17.06.2021	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»		*Виконано*
8.	Виконання завдання до підрозділу «Основи хорони	17.06.2021	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	18.06.2021	*Виконано*
10.	Нормоконтроль	19.06.2021	*Виконано*
11.	Перевірка на плагіат	19.06.2021	*Виконано*
12.	Попередній захист кваліфікаційної роботи	19.06.2021	*Виконано*
13.	Захист кваліфікаційної роботи	23.06.2021	

Студент _____ Суховерша В. О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-сервісу для моніторингу роботи серверного обладнання // Кваліфікаційна робота освітнього рівня «Бакалавр»// Суховерша Володимир Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41// Тернопіль, 2021 // сторінок – 59, рисунки– 40, таблиць –1, джерел –40.

Ключові слова: веб-сервіс, серверне обладнання, системи Nagios, технології моніторингу серверного обладнання, програмно-апаратні засоби.

У кваліфікаційній роботі розроблено архітектуру програмного додатку веб-сервісу для моніторингу роботи серверного обладнання, що дозволить. Створено та описано структурні схеми, основні компоненти. Розглянуто функціональну структуру системи та її основних елементів – модулів обробки даних. Визначено основні елементи бази даних та встановлено зв'язки між ними. Спроектовано структуру бази даних.

Здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. Розкрито питання встановлення та налаштування програмного забезпечення на сервері, а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

ANNOTATION

Web-service development for server equipment operation monitoring // Qualification work of educational level "Bachelor" // Sukhoversha Volodymyr Oleksandrovych // Ivan PuliyuTernopil National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science,group. CHc-41 // Ternopil, 2021 // pages – 59, figures– 40, tables–1, sources–40.

Keywords: web service, server equipment, Nagios systems, server equipment monitoring technologies, software and hardware.

In the qualification work, the architecture of the web service software application was developed to monitor the operation of the server equipment, which will allow. Structural schemes, main components are created and described. The functional structure of the system and its main elements - data processing modules are considered. The main elements of the database are identified and the links between them are established. The database structure is designed.

The description of testing procedures and their results is described, the test requirements to the software are described, and also defects are revealed. The issues of installing and configuring the software on the server are revealed, as well as the requirements that must be met to use the system, the user instructions for working with the system are described.

ЗМІСТ

ВСТУП

На сьогоднішній день всі сучасні установи мають локальну мережу, що складається з персональних комп'ютерів, мережевого обладнання, серверів і шлюзу для доступу в інтернет. Також більшість установ, поряд з компаніями і підприємствами, постійно розвивають свою мережеву інфраструктуру, додаючи сервера та мережеве обладнання для створення додаткових інформаційних ресурсів.

Створення комплексної системи управління серверами приносить ряд незаперечних переваг при експлуатації Дата-центрів. Складається повне розуміння, які сервери і в яких цілях використовуються, чи справні вони. На всіх серверах стоять відомі версії ОС і додатків, з відомими наборами патчів. Якщо в ПЗ виявляється серйозна помилка або вразливість, досить додати потрібний патч в політику, і він автоматично встановлюється туди, де необхідний. Для спостереження за роботою цієї структури та її технічного обслуговування необхідно використання системи централізованого моніторингу. Система моніторингу може забезпечити цілодобовий контроль всіх ресурсів і своєчасно оповістити про порушення роботи даних ресурсів.

Метою даної роботи є проектування та практична реалізація системи управління серверним обладнанням та її адаптація через iLO в Nagios.

Для досягнення зазначеної мети необхідно вирішити такі завдання:

1) вивчення та аналіз особливостей існуючих систем моніторингу серверного обладнання;

2) вивчення особливостей роботи з платформами Nagios та Nagvis;

3) розробка архітектури web-сервісу моніторингу серверного обладнання;

5) розробка користувацького інтерфейсу web-сервісу моніторингу серверного обладнання;

6) реалізація web-сервісу моніторингу серверного обладнання;

7) тестування web-сервісу моніторингу серверного обладнання.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПРОЕКТУВАННЯ WEB-СЕРВІСУ МОНІТОРИНГУ СЕРВЕРНОГО ОБЛАДНАННЯ

1.1. Задачі моніторингу серверного обладнання та порівняльний аналіз відомих програмно-апаратних засобів моніторингу серверного обладнання

Засоби і системи моніторингу серверного обладнання розвивалися в міру збільшення числа серверів у складі IT-інфраструктури підприємств та організацій. У перший час застосування цих систем було спрямовано, насамперед, на забезпечення автоматизації рутинних завдань і підвищення ефективності роботи системних адміністраторів. Сьогодні проблема управління серверами стає більш комплексною і переростає в завдання підвищення ефективності використання IT в цілому на підприємстві. Постають питання окупності інвестицій в IT-сферу, перекладу IT-підрозділів на сервісно-орієнтовану модель роботи, застосування можливостей аутсорсингу. І, як і раніше, залишається завдання забезпечити безперебійну роботу серверів і прикладних додатків, виключити втрату даних, скоротивши при цьому витрати на експлуатацію інфраструктури.

Можна виділити такі загальні завдання управління серверною інфраструктурою в сучасному центрі обробки даних: класичне адміністрування; облік і контроль наявних ресурсів; адаптація інфраструктури до потреб додатків; автоматизація управління програмним забезпеченням.

Під класичним адмініструванням розуміється індивідуальне адміністрування кожного сервера незалежно від інших, доступ до консолі сервера, управління електроживленням тощо (рисунок 1.1). Сучасні системи управління серверами забезпечують віддалене виконання адміністратором цих функцій. Така можливість може бути реалізована як програмними, так і апаратними засобами.

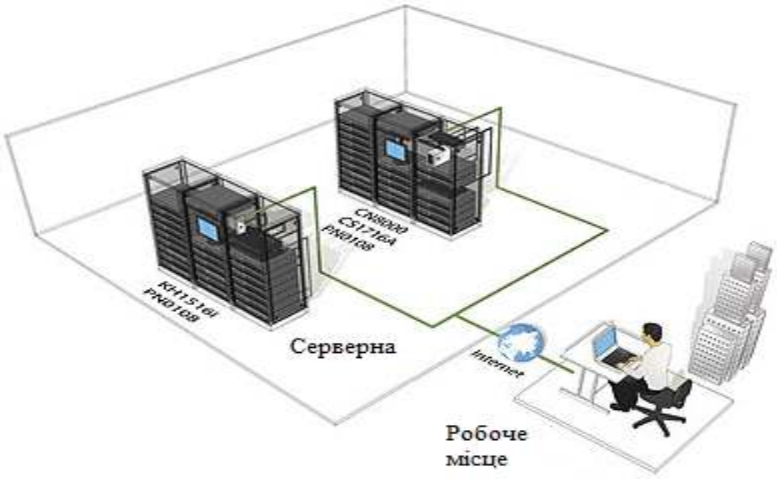

Рисунок 1.1 - Схема віддаленого управління сервером

Віддалене управління серверами HP здійснюється за допомогою iLO (Integrated Lights-Out), в молодших серіях серверів - Lights-Out 100 (LO100i). У новому поколінні серверів ProLiant Gen8 абревіатура iLO склалася з інших слів - Insight Lifecycle Onboard - і отримала версію 4. iLO 3 являє собою RISC-процесор з частотою 250 МГц зі своєю оперативною пам'яттю 128 МБ DDR 3 ECC (400 МГц і 256 МБ у версії 4), вбудованими в материнську плату сервера, і мережевим інтерфейсом зі швидкістю 10/100 Мбіт/с, що розділяються з інтерфейсом загального призначення або виділеним (у деяких моделях серверів, що не мають виділеного порту, окремий порт можна отримати за допомогою спеціальної карти розширення), тобто є таким собі мікрокомп'ютером в сервері.

iLO призначений для надання віддаленого доступу до сервера, а також виконує функції моніторингу стану сервера та обмеження сервера по споживані енергії (якщо це необхідно). Материнські плати серверів ProLiant G7 містять на собі 32 датчика температури, Gen8 - 64. Така кількість дозволяє встановити датчики біля будь-якого компонента, схильного до відчутного нагрівання, щоб при необхідності його охолодження збільшувати обороти тільки вентилятора, що знаходиться найближче до потенційної точці перегріву, і тим самим знижувати споживану енергію в порівнянні з випадком, коли для охолодження сервер розганяє всі свої вентилятори. Моніторинг датчиків і керування вентиляторами покладені на iLO.

Рисунок 1.2 - HP iLO 3

Вбудоване в сервер програмне забезпечення, що забезпечує функціонал iLO, відрізняється для серверів ProLiant і для блейд-систем. За назвою їх легко відрізнити : наприклад , iLO 3 і iLO 3 for BladeSystem.

Advanced розширює функціонал до максимально можливого. До набору Standard він додає повноцінну консоль, діючу завжди - образів і папок з керуючого комп'ютера як віртуальних носіїв до сервера і можливістю завантаження з них, можливість запису і подальшого програвання відбувається на моніторі, Terminal Services (підключення до iLO при завантаженій Windows через службу терміналів для збільшення швидкості роботи консолі), оптимізацію споживаної потужності, аутентифікацію через Active Directory, груповий доступ до консолі.

Щоб підключитися до iLO, спочатку необхідно призначити IP- адресу його інтерфейсу. Далі використовуємо призначену вручну або за DHCP адресу. Надалі адресу можна міняти ще і з Web- інтерфейсу (рисунок 1.3).

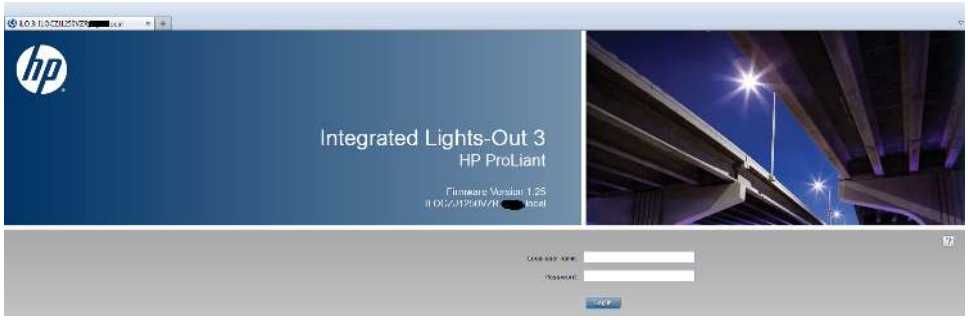

Рисунок 1.3 - HP iLO 3

У системі можливо переглянути детальну інформацію про процесори, які використовуються на сервері (рисунок 1.4):

Processor 1

Processor Speed	2667 MHz
Execution technology	4/4 cores; 8 threads
Memory Technology	64-bit Capable
Internal L1 cache	128 KB
Internal L2 cache	1024 KB
Internal L3 cache	12288 KB

Processor 2

Processor Speed	2667 MHz
Execution technology	4/4 cores; 8 threads
Memory Technology	64-bit Capable
Internal L1 cache	128 KB
Internal L2 cache	1024 KB
Internal L3 cache	12288 KB

Рисунок 1.4 - Інформація про процесори в iLO 3

В iLO Event Log зберігається інформація про події iLO (рисунок 1.5). Корисно при розслідуваннях - хто заходив на iLO, які операції виконувалися.

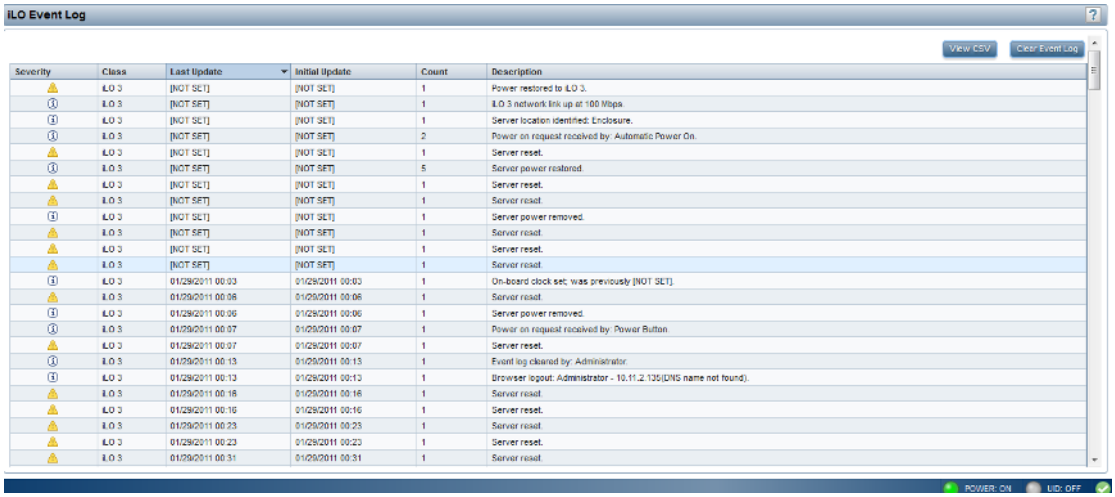

Рисунок 1.5 - Операція логування в iLO 3

Integrated Management Log (IML) містить відомості про проблеми і події сервера (рисунок 1.6):

Integrated Management Log

Severity	Class	Last Update	Initial Update	Count	Description
⚠	POST Message	01/29/2011 01:03	01/29/2011 01:03	1	POST Error: 1785-Drive Array not Configured
ⓘ	Maintenance	01/29/2011 00:16	01/29/2011 00:16	1	Maintenance note: IML cleared through hpasmcli
⚠	POST Message	01/29/2011 00:03	01/29/2011 00:03	1	POST Error: 1794-Drive Array - Array Accelerator Battery Charge Low

Рисунок 1.6 - Integrated Management Log

Diagnostics - інформація про сам iLO , результати самотестування. Тут можна перевантажити iLO (рисунок 1.7).

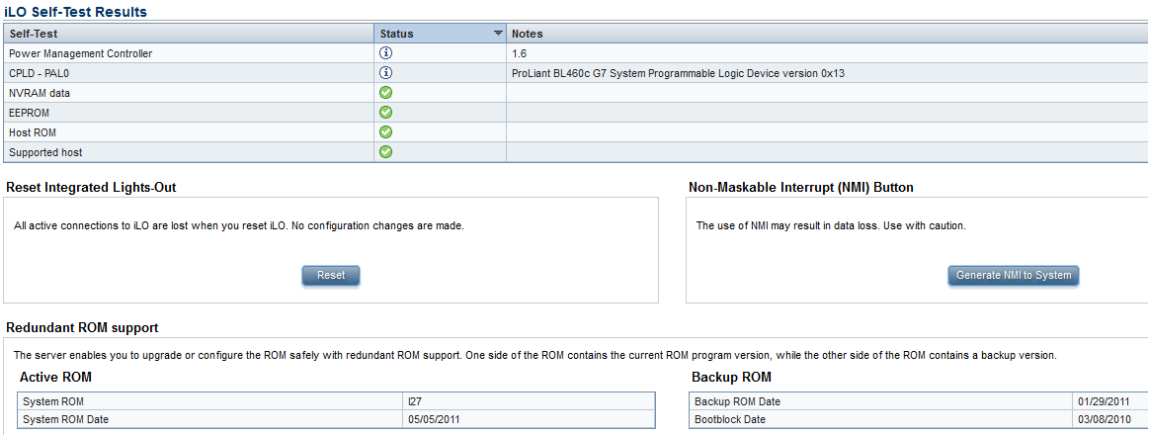

Рисунок 1.7 - Diagnostics в iLO 3

Далі - один з найважливіших пунктів - Remote Console (рисунок 1.8-1.9). З цієї сторінки можна запустити віддалену консоль на сервері. iLO має два види консолей: що працюють на . NET Framework (на машині адміністратора буде потрібно версія не нижче 3.5) і Java.

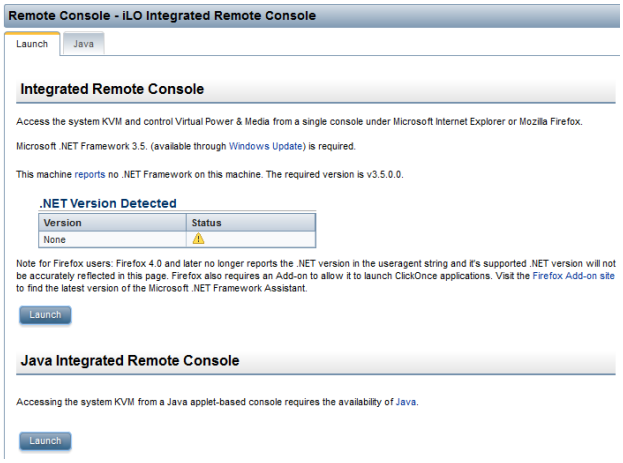

Рисунок 1.8 - Виклик Remote Console в iLO 3

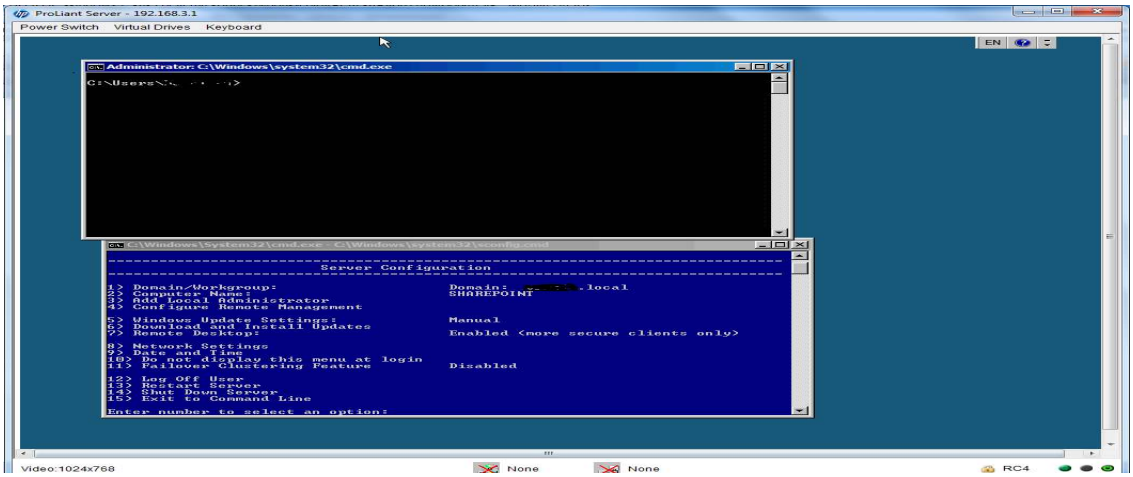

Рисунок 1.9 - Remote Console в iLO 3

Через цю консоль можна працювати з самим сервером: ви начебто сидите за монітором, клавіатурою і мишкою, підключеними прямо до машини. Можна керувати живленням сервера: віртуально натиснути на кнопку включення (Momentary Press), що включить вимкнений сервер і вимкне включений, при чому в залежності від ОС вимикання може бути коректним з точки зору ОС; форсовано вимкнути сервер утриманням кнопки на 6 секунд (Press and Hold) - корисно, якщо ОС не відгукується на Momentary Press, але некоректно; вимкнути і включити (з затримкою приблизно 6 с.) одноразовим натисканням (Cold Boot); перезавантажити сервер (Reset).

Power Management - Server Power. Звідси можна виконати з живленням ті ж дії, що і через Remote Console (при цьому контролювати стан сервера через індикатор «System Power»), а також задати автоматичне включення сервера при подачі живлення і налаштувати затримку запуску (рисунок 1.10).

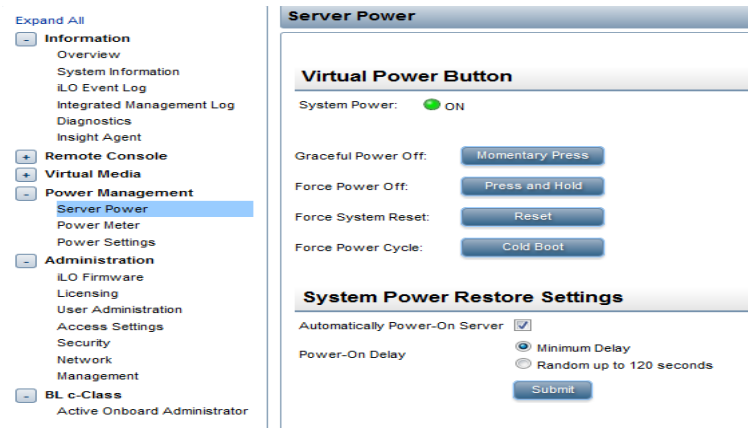

Рисунок 1.10 - Power Management - Server Power.

Power Meter. У цій вкладці відображається споживана зараз потужність, ліміт споживання, історія споживання за останні 24 години і 20 хвилин (рисунок 1.11).

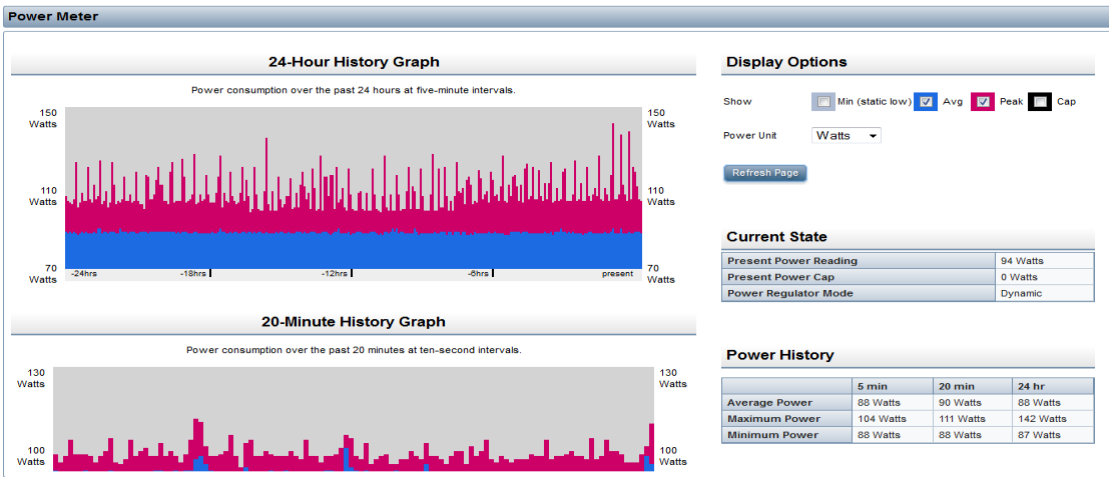

Рисунок 1.11 - Power Management - Server Power

Апаратно iDRAC 6 містить процесор з тактовою частотою 220 МГц , що використовує 128 МБ оперативної пам’яті сервера. Розташовується він на платі Express (рисунок 1.12), адже на відміну від iLO, iDRAC не вбудовується в материнську плату, він являє собою окремий компонент. Це маленька плата з мікросхемами, що купується як опція до сервера і вставляється в спеціальний порт на материнській платі, причому для iDRAC 6 Express і Enterprise - це дві різні плати. Також iDRAC 6 Enterprise може мати ще й карту пам'яті SD об'ємом 1 ГБ (технологія vFlash, карту можна використовувати як завантажувальний для діагностики пристрій).

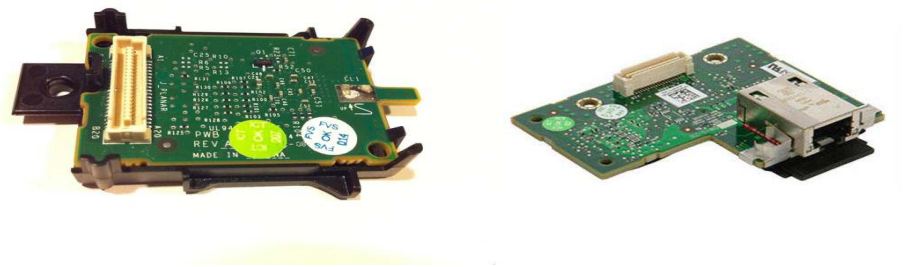

Рисунок 1.12 - Карта iDRAC 6 Express

Плата iDRAC 6 Express коштує близько 60$, Enterprise - 150$. У цьому аспекті Dell краще HP захистилася від некупівлі ліцензії: ліцензію на iLO Advanced можна пошукати в Інтернеті, а деталь доведеться купити.

Щоб підключитися до iDRAC, спочатку необхідно призначити IP- адресу його інтерфейсу. За замовчуванням iDRAC працює за адресою 192.168.0.120. Детальна інформація про сервер (рисунок 1.13):

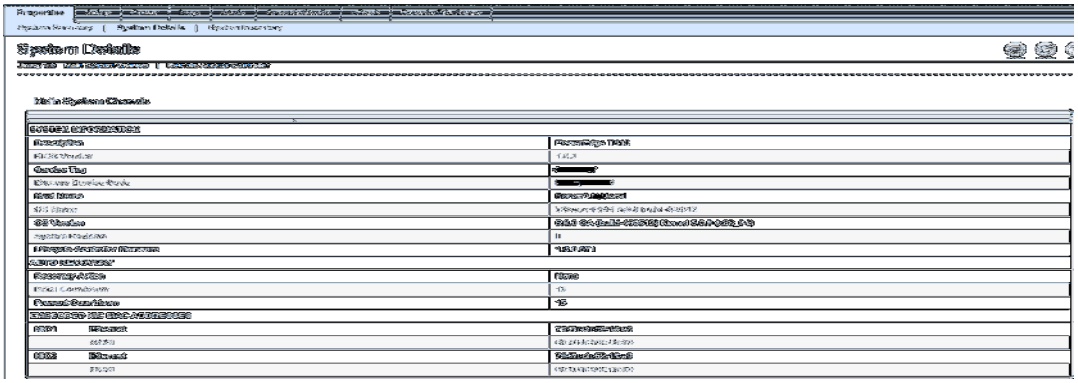

Рисунок 1.13 - Детальна інформація про сервер

Можливість збереження логів на віддаленому сервері за стандартом syslog:

1.2. Аналіз відомих програмних технологій моніторингу серверного обладнання с*пецифікація вимог до системи*

Система моніторингу - це потужний засіб підтримки і аналізу стану ресурсів, що дозволяє забезпечувати цілодобовий контроль за мережевими ресурсами масового користування. Використання такої системи дозволяє відстежити можливі збої і причини виходу з ладу мережевого обладнання. Вона також забезпечує оповіщення про критичний стан ресурсів, що дозволяє негайно вжити заходів для підтримки роботи цих ресурсів.

Nagios, спочатку створена під ім'ям Netsaint, розроблена Етаном Галстадом (Ethan Galstad). Він же підтримує і розвиває систему сьогодні, спільно з командою розробників, які займаються як офіційними, так і неофіційними плагінами.

Спочатку Nagios була розроблена для роботи під GNU / Linux, але вона також добре працює і під іншими Unix-подібними ОС. Nagios поширюється по ліцензії GNU General Public License Version 2.

Основна відмінність Icinga від Nagios:

• власний інтерфейс;

• модифікована структура взаємодії API і ядра системи.

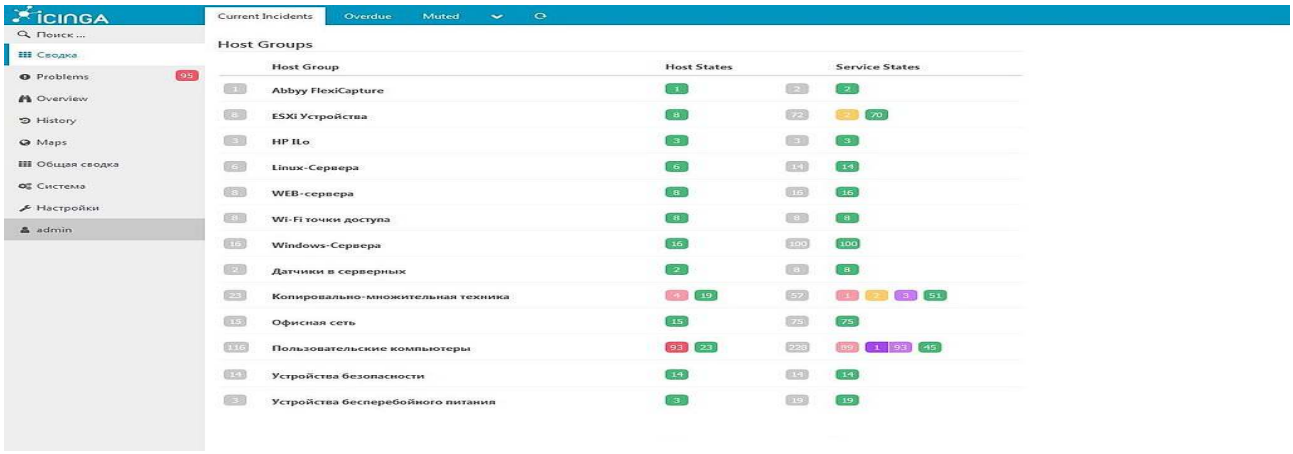

Рисунок 1.14 - Система Icinga

Система Big Sister (рисунок 1.15) - це системний і мережевий монітор з відкритим кодом, з ліцензією GNU. Спочатку був тільки системою мережевого моніторингу, а тепер є мультипроект в області систем управління і моніторингу. У неї входить:

• монітор мережевих систем (серверів);

• надання легкого відстеження в реальному часі стану мережі;

• попередження про критичний стан систем;

• генерація історії змін станів;

• збір і відображення різних даних по продуктивності систем;

• зберігання зібраних даних в MySQL або текстовому файлі.

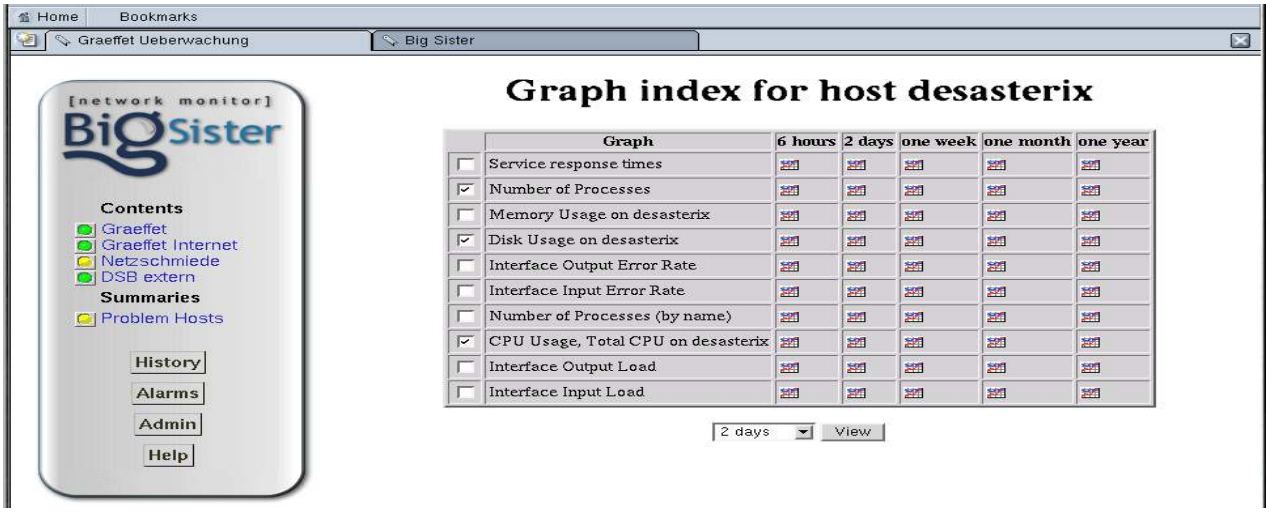

Рисунок 1.15 - Система Big Sister

Система Ganglia - масштабована розподілена система моніторингу для високопродуктивних систем, таких як кластерні і грід системи. Активно використовуються такі технології:

• XML - для передачі даних;

• XDR для компактного, портативного транспорту даних;

• RRDtool для зберігання даних і візуалізації.

Pandora FMS можуть бути розгорнуті практично в будь-якій операційній системі. Моніторинг здійснюється засобами (WMI, SNMP, TCP. UDP, ICMP, HTTP) і агентів. Агенти доступні для кожної платформи. Вона може також контролювати апаратні системи з TCP/IP стеком такі, як балансування навантаження, маршрутизатори, мережеві комутатори, принтери і брандмауери.

Можливості:

• виявлення нових систем в мережі;

• перевірка на наявність або продуктивність;

• дозволяють отримувати дані всередині систем зі своїм агентам Lite (майже кожна операційна система);

• дозволяють отримувати дані із зовні, використовуючи тільки зонди мережі.;

• створення в реальному часі звітів і графіків;

• висока доступність для кожного компонента;

• масштабованість і модульна архітектура;

• підтримує до 2500 модулів на сервері;

У процесі аналізу встановлено, що не всі розглянуті системи в повній мірі покривають необхідний функціонал. Для систем, що покривають необхідний функціонал, складена таблиця, яка відображає основні критерії вибору системи для розроблюваного рішення.

1.3. Розроблення архітектури програмної системи п*роектування структури бази даних*

Користувач взаємодіє з системою за допомогою web-браузера. Як сховище даних виступає СУБД MySQL, а сполучною ланкою браузера користувача і сховища є web-сервер додатків Tomcat, який здійснює динамічну генерацію сторінок з використанням пакета KemsuWeb. Така архітектура дозволяє працювати з системою з будь-якої точки, що має вихід в Інтернет, а також забезпечує незалежність від платформи і мінімальні системні вимоги до апаратного забезпечення користувача. Менеджер процесів моніторингу забезпечує передачу агентам завдань для здійснення процесів тестування та діагностики серверів, а також здійснює прийом результатів моніторингу і розміщення їх у сховище даних.

Взаємодія менеджера моніторингу з базою даних здійснюється через спеціальні процедури, призначені для виконання операцій з базою даних (наприклад, для входу в систему, розбору і виконання SQL-запитів, отримання записів тощо).

Сховище даних містить файли і метаінформацію, необхідну для організації проведення моніторингу та статистичного аналізу. Система віддаленого доступу до інтерфейсу системи моніторингу, тобто клієнтська частина даної системи побудована за трьохрівневою архітектурою «клієнт - сервер додатків - сервер баз даних» (рисунок 1.16) і взаємодіє з системою моніторингу на рівні загальної бази даних.

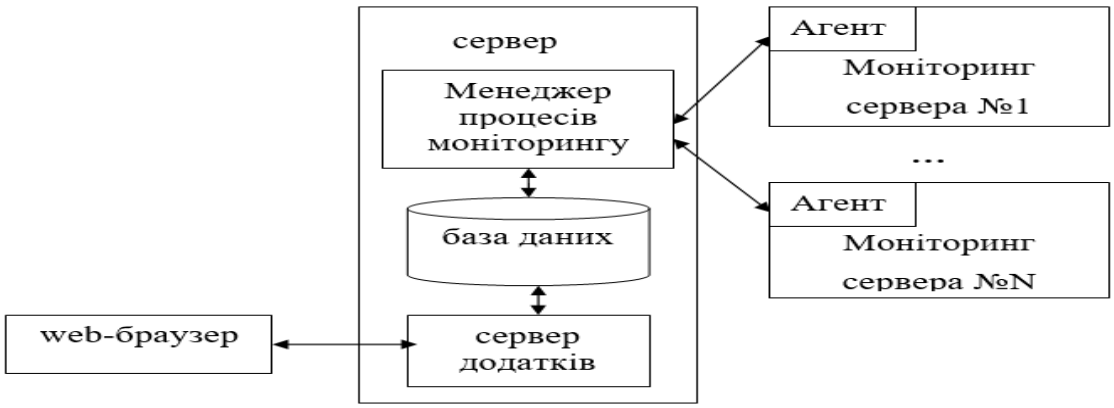

Рисунок 1.16 - Структура системи моніторингу

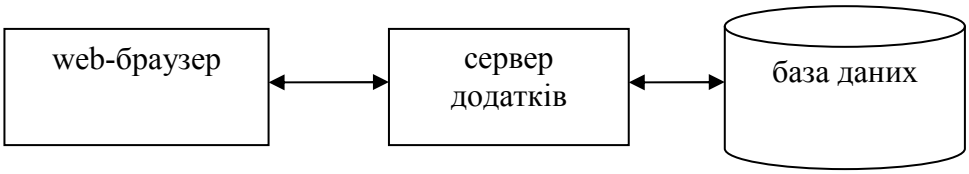

Рисунок 1.17 - Структура системи віддаленого доступу

За допомогою клієнтського додатка (web-браузера) користувач переглядає поточний стан об’єктів в базі даних і може створювати нові об’єкти, змінювати параметри вже існуючих і видаляти об’єкти, викликаючи процедури, збережені на сервері баз даних.

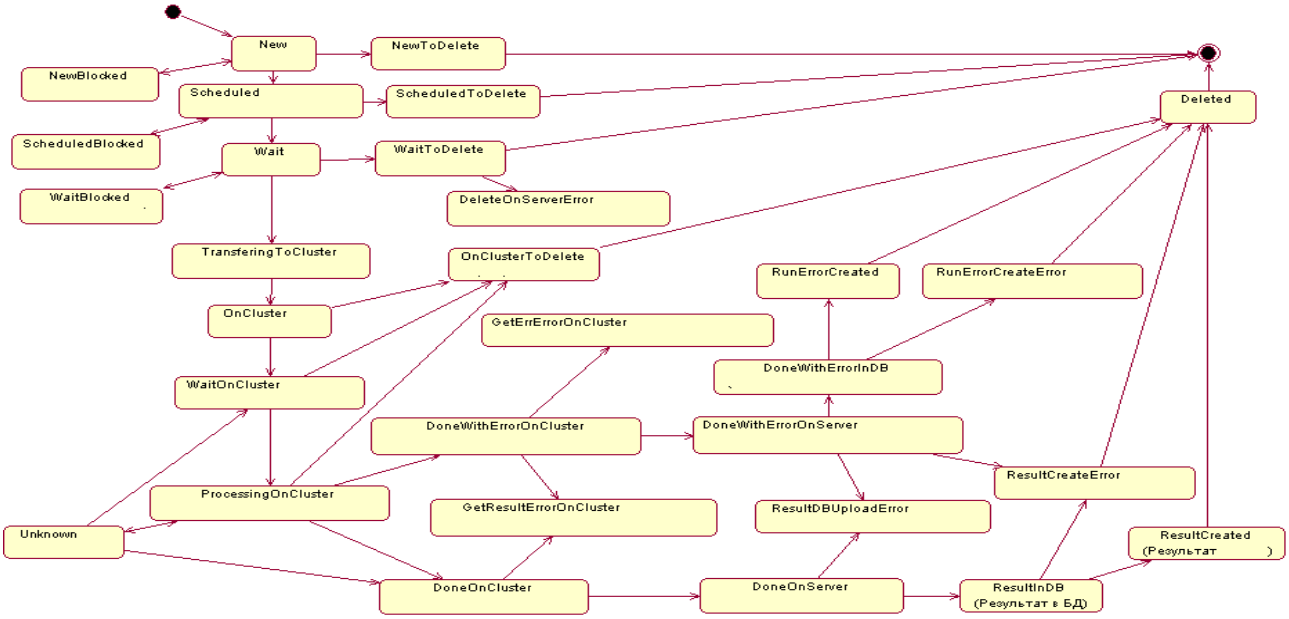

Рисунок 1.18 - Діаграма станів процесу моніторингу

Логіка системи реалізується на сервері додатків і сервері баз даних. Як сервер додатків використовується Apache, призначений для реалізації технології з використанням Python і виконує функції web-сервера. В якості системи управління базою даних був використаний MySQL. В якості клієнта до системи використовується стандартний web-браузер, обмін даними проходить за стандартними мережевим протоколам в середовищі Intranet/Internet.

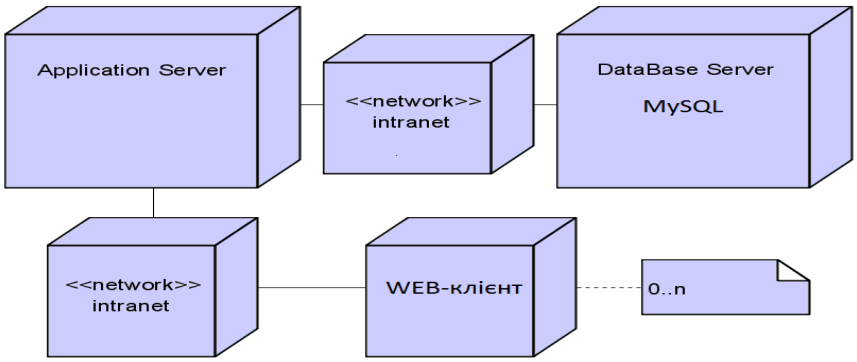

Рисунок 1.19 - Архітектура системи на фізичному рівні

Логіка програми на стороні сервера бази даних реалізована у вигляді набору пакетів: для функціональних блоків створюються пакети для здійснення операцій з даними, що забезпечують маніпуляцію необхідними даними шляхом виклику збережених процедур. Для реалізації пакетів використовується мова PL/SQL. Для реалізації web-інтерфейсу використовуються технології J2EE і XML.

На сервері додатків Apache використана бібліотека шаблонів, яка забезпечує єдине середовище для створення додатків, заснованих на трирівневій архітектурі в середовищі Internet за рахунок адаптерів, які задовольняють різні потреби розробника: в операціях з MySQL, у захисті інформації, в управлінні ходом виконання програми.

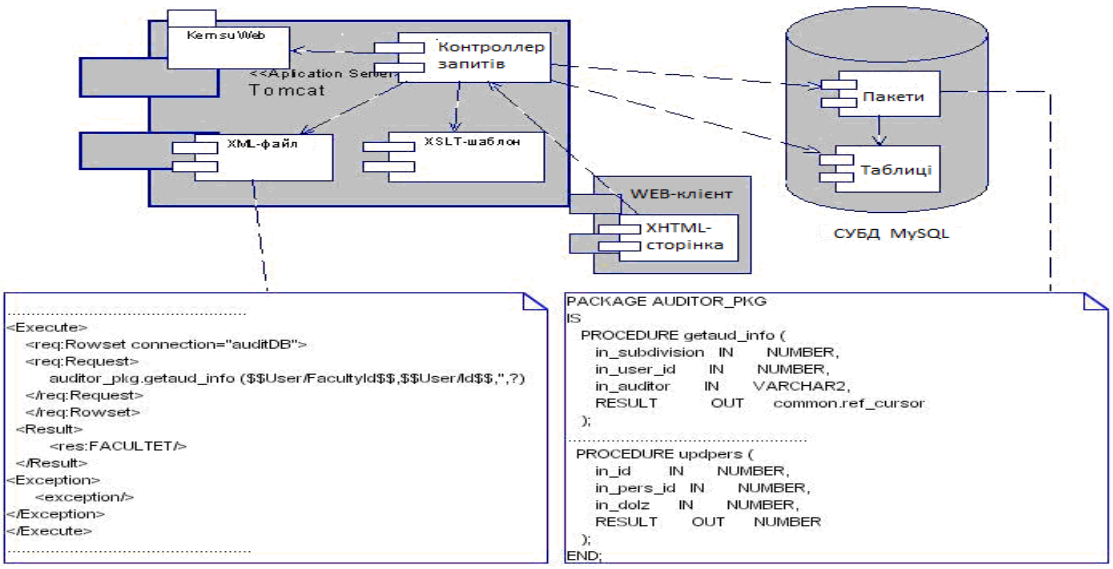

Рисунок 1.20 - Архітектура реалізації

Розробник створює XSLT - шаблони оформлення інтерфейсу користувача і XML - файли, в яких описується виклик збережених процедур і обробка отриманих результатів з використанням спеціальних «команд», які описуються елементами з простору імен.

Web-клієнт надсилає HTTP-запит RequestController. Додаток витягує параметри з HTTP-запиту, включаючи URL, тобто адреса XML -файла, який надалі перетворюється в абсолютний шлях до файлу. Скрипт створює екземпляр класу XMLTraslator, передаючи йому HTTP-запит і шлях до файлу. Якщо в XML-файлі зустрічаються елементи зі спеціального простору імен, створюється адаптер і йому делегується обробка таких елементів. З XML-файла витягується шлях до шаблону трансформації XSLT main.xsl. Цей шаблон в свою чергу імпортує шаблон default.xsl, який забезпечує механізм простого оформлення результатів запитів у відповідь на "команди", описувані в XML-файлі і tree.xsl, який містить структуру WEB-додатку та шаблони-функції для різних дій з цією структурою. На завершення main.xsl динамічно оформляє XHTML-файл, який повертається web-клієнту за допомогою скрипта (рисунок 1.21).

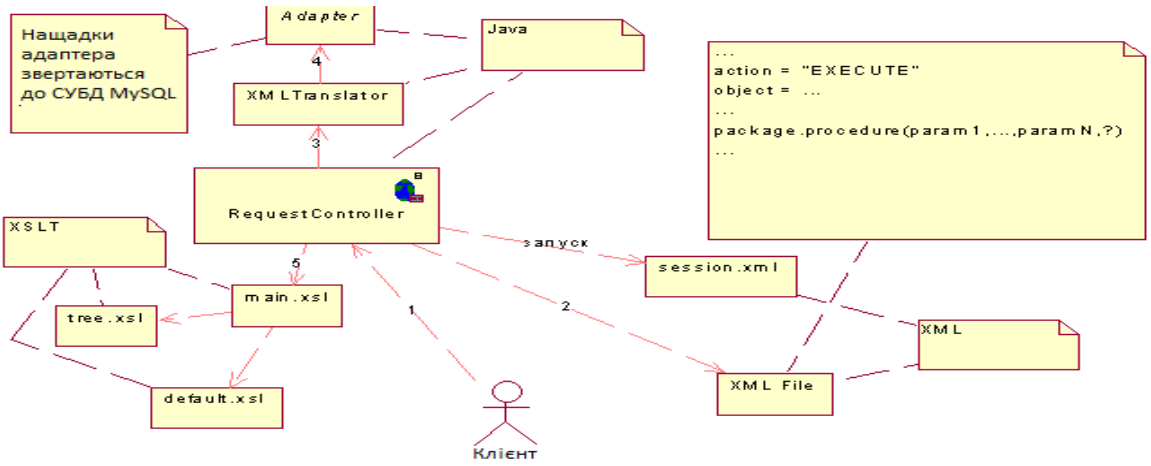

Рисунок 1.21 - Архітектура web-сервісу

ER-модель бази даних системи моніторингу серверного обладнання містить 37 сутностей із загальною кількістю атрибутів понад 200. Повна ER-модель бази даних наведена на рисунку

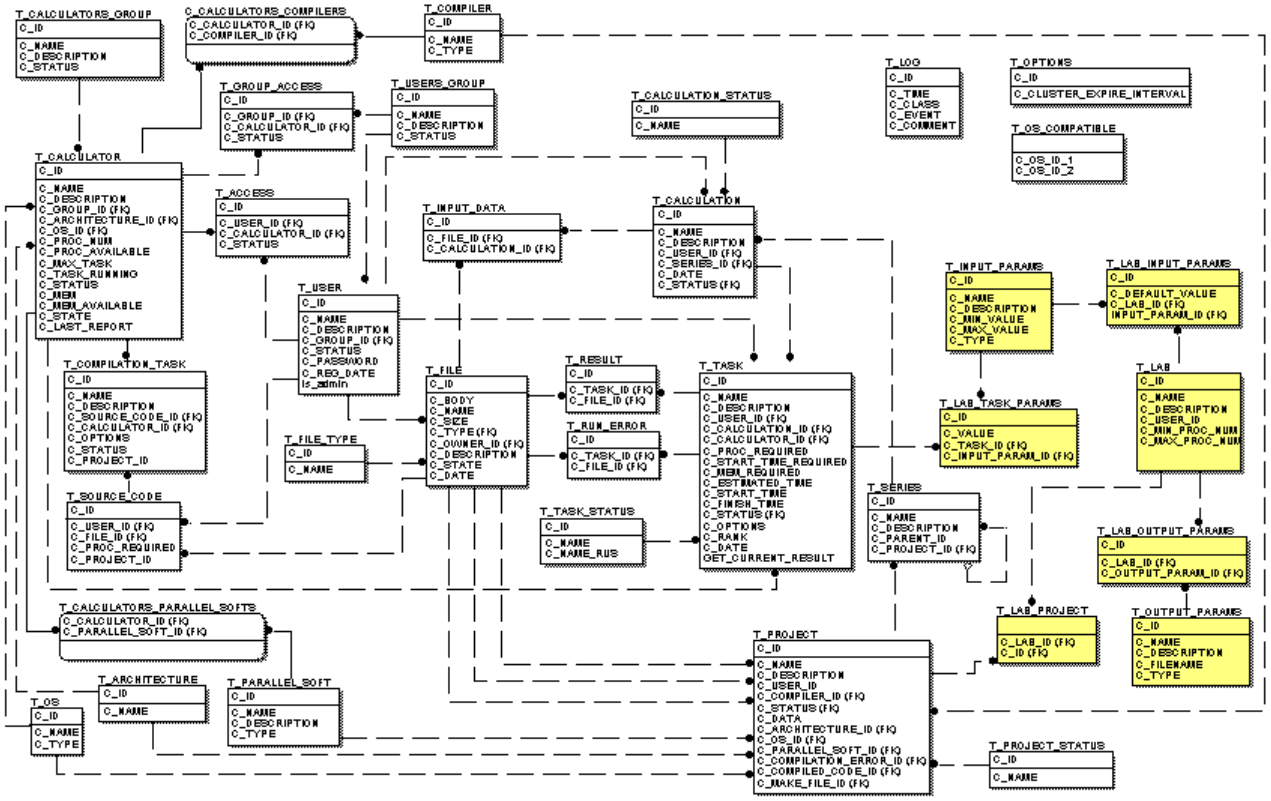

Рисунок 1.22 - ER-модель БД системи моніторингу серверного обладнання

Висновки до розділу 1

Здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих методів позиціонування сайтів. Здійснено аналіз вимог до програмної системи.

У цьому розділі розроблено архітектуру програмного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних. Визначено основні елементи бази даних та встановлено зв’язки між ними. Спроектовано структуру бази даних.

РОЗДІЛ 2

ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-СЕРВІСУ МОНІТОРИНГУ СЕРВЕРНОГО ОБЛАДНАННЯ, ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

2.1. Встановлення та базове налаштування системи Nagios

Для розробки web-сервісе моніторингу серверного обладнання були використані наступні технології:

- мова програмування Python 3 (рисунок 2.1).

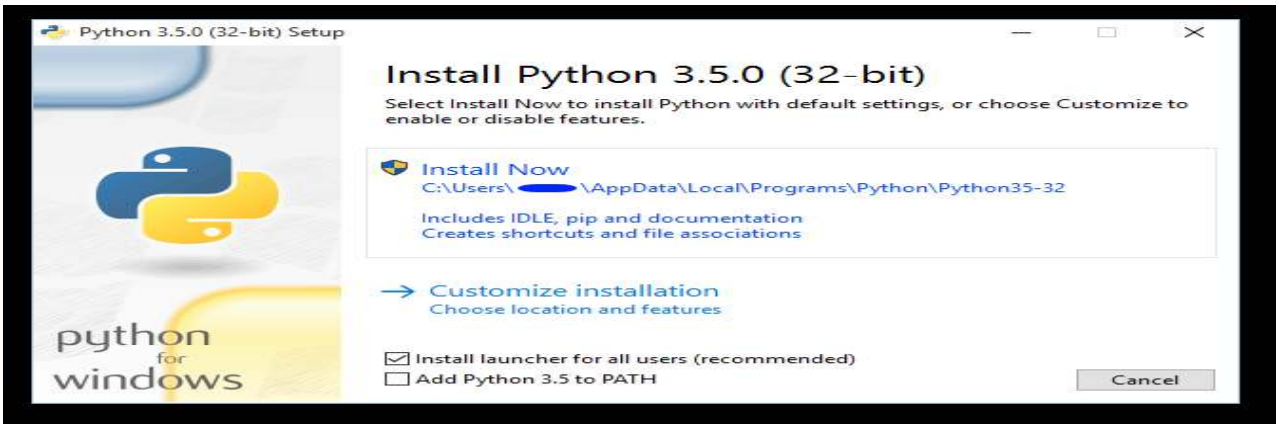

Рисунок 2.1 - Мова програмування Python 3

Python був розроблений в кінці вісімдесятих і на початку дев'яностих оків в Національному науково-дослідному інституті математики та інформатики в Нідерландах.

Python є похідним від багатьох інших мов, в тому числі ABC, Modula-3, C, C ++, Алгол-68, Smalltalk і Unix оболонки і інших мов сценаріїв.

Python є авторським правом. Як Perl, Python вихідний код доступний під ліцензією GNU General Public License (GPL).

Python тепер підтримується командою розробників ядра в інституті.

Ця програма установки запропонує також встановити додаткові пакети, необхідні для системи Nagios. У ці доповнення входять в пакет nagios-plugins,

що містить набір програм і тестів, для збору даних про стани локальних ресурсів сервера і для визначення роботи мережевих служб.

Для працездатності системи на сервері також повинен бути встановлений web-сервер, в даному випадку ми використовуємо Apache. Пакет для установки web -сервера Apache також знаходиться в репозиторії, тому для його установки необхідно виконати команду: emerge apache.

Після установки web -сервера і системи моніторингу запускаємо їх, для перевірки. Всі конфігураційні файли системи знаходяться в директорії /etc/nagios/. Основний конфігураційний файл nagios.cfg. У цьому файлі конфігурації містяться дані за місцем розташування логів, ім'я та група користувача від якого запускається Nagios, дані про розташування конфігураційних файлів: мережевих вузлів, команд перевірок, груп. У конфігураційному файлі nagios.cfg потрібно вказати:

```
log_file = / var / nagios / nagios.log - основний лог системи Nagios
cfg_file = / etc / nagios / objects / services.cfg - конфігураційний файл з описом сервісів, які
повинні перевіряти необхідні мережеві служби та локальні ресурси серверів.
cfg_file = / etc / nagios / objects / commands.cfg - конфігураційний файл з командами
cfg_file = / etc / nagios / objects / contacts.cfg -
cfg_file = / etc / nagios / objects / timeperiods.cfg -
cfg_file = / etc / nagios / objects / templates.cfg -
cfg_file = / etc / nagios / objects / hosts.cfg - конфігураційний файл з хостами
cfg_file = / etc / nagios / objects / groups.cfg - конфігураційний файл з групами
nagios_user = nagios
nagios_group = nagios
```

У файлі commands.cfg можна створювати власні команди і модифікації існуючих:

```
define command{
        command_name    check_nodata
        command_line    $USER1$/check_nodata
        }

define command{
        command_name    check_smb_unix
        command_line    $USER1$/check_tcp -H $HOSTNAME$ -p 445
        }
```

У файлі groups.cfg містяться групи, на які розділені хости:

```
define hostgroup{
        hostgroup_name  Multimedia ; The name of the hostgroup
        alias           Multimedia Rescues; Long name of the group
        members         div0.auditory.ru, div1.auditory.ru, dvr.aditory.ry,
dsip.auditory.ru
        }

define hostgroup{
        hostgroup_name  ICT; The name of the hostgroup
        alias           ICT Rescues; Long name of the group
        members         www.auditory.ru, forum.auditory.ru,
share.auditory.ru, lms.auditory.ru
        }
```

Файл services.cfg містить параметри активних і пасивних перевірок мережевих служб і локальних ресурсів мережевих хостів.

```
define service {
        service_description     Mem Free
        check_command   check_nodata
        use     passive-service
        host_name       dhl.auditory.ru
        contact_groups  admins
}

define service {
        check_command   check_ssh
        service_description     SSH
        process_perf_data       0
        use     active-service
        host_name       dhl.auditory.ru
        contact_groups  admins
}
```

Крім налаштування системи безпосередньо через конфігураційні файли пропонується використовувати компонент ym, який надає набір команд для зручної і швидкої настройки системи.

Для того щоб подивитися опис команд, можна виконати команду ym з специфікатором - h, також цей специфікатор можна використовувати на всіх етапах складання команди налаштувань:

```
ym -h  (выведет весь список команд)
...
или
ym -c add-host -h (выведет список параметров для команды )

Usage: ./ym -c COMMAND
add-host -  Add one or more hosts.
        -o hostname|READ_FILE=file [-o object2 -o object3]
        -p use=default-host -p contact_groups=some-admins


ym -c add-service -h

Usage: ./ym -c COMMAND
add-service -  Add service(s) to host(s)
        -o service|READ_FILE=file [-o object2 -o object3]
```

В процесі виконання роботи розроблений інструмент, який дозволяє швидко додавати групу хостів з усіма сервісами. Для його використання необхідно спочатку встановити компонент по збору даних на нові хости. Даний

інструмент працює у зв'язці з інструментом ym і компонентом по збору даних, згаданому раніше. Цей інструмент можна викликати командою: hdtp – hosts.

Після чого інструмент перевірить, чи з'явилися на сервері дані про нові хости, на яких був встановлений компонент по збору даних. Якщо дані про нові хости є, то інструмент hdtp-hosts запропонує додати нові хости з усіма його сервісами.

В результаті налаштувань на сторінці Nagios->Service Detail можна побачити стан хостів та їх сервісів (рисунок 2.2).

Рисунок 2.2 - Результат моніторингу в системі Nagios.

Для налаштування оповіщення по email насамперед необхідно налаштувати відправку email повідомлень через командний рядок. Для цього необхідно встановити програму, яка повинна передавати листи на поштовий сервер. Наприклад можна встановити пакет mail-mta/ssmtp [13], це можна зробити командою: emerge mail-mta/ssmtp.

Далі необхідно налаштувати конфігураційний файл /etc/ssmtp.conf. У ньому необхідно вказати поштову адресу, з якої будуть відправлятися листи, а також поштовий сервер, тип з’єднання і авторизаційні дані. Наприклад, для поштової скриньки на gmail.com

```
# / Etc / ssmtp.conf - конфігураційний файл sSMTP sendmail.
#
mailhub = smtp.gmail.com: 465

hostname = Nagios
FromLineOverride = YES
```

Рисунок 2.3 - Налаштування конфігураційного файлу

Тепер можна відправляти листи використовуючи команду /usr/bin/mail. Далі необхідно створити контакт для системи моніторингу або модифікувати існуючий, приклад створення нового контакту:

```
define contact {
   contact_name nagiosadmin
   alias NagiosAdmin
   service_notification_period 24x7
   host_notification_period 24x7
   service_notification_options w,u,c,r
   host_notification_options d,r
   service_notification_commands notify-service-by-email
   host_notification_commands notify-host-by-email
```

Рисунок 2.4 - Налаштування конфігураційного файлу

Даний метод, по суті є розширенням для оповіщення по електронній пошті. А саме, оповіщення Nagios посилає повідомлення на електронну пошту, а адміністратор, підключивши послугу мобільного оператора, може отримати SMS про новий лист і короткий його зміст. Даний метод не є найбільш зручним, але може бути корисним якщо немає іншої можливості оповістити по SMS.

Для конкретної реалізації даного сервісу використовувалися такі засоби:

• мобільний телефон Nokia;

• програма для управління телефоном засобами консолі gammu;

• USB кабель.

Для використання мобільного телефону як GPRS модему і відправки SMS оповіщення, необхідно включити в ядрі системи підтримку:

```
Device Drivers--->[*] USB Support --->
  <*> USB Modem (CDC ACM) support
```

Для установки програми потрібно виконати команду: emerge gammu.

Для налаштування системи можна скористатися майстром налаштувань, виконавши команду: gammu – config.

В даному майстрі потрібно вказати наступні дані:

• Port / dev/ttyACM0;

• Connection (at19200 - тип драйвера для мобільного телефону Nokia);

• Model (NAUTO);

• Synchronize time yes ;

• Log file/var/log/log_name.log .

Для використання функції SMS оповіщення в системі Nagios необхідно створити команду, в конфігураційному файлі команд commands.cfg.

```
# 'notify-host-by-sms' command definition
define command{
  command_name notify-host-by-sms
  command_line /usr/bin/printf "%b" "** Nagios **\n\nNotification Type:
$NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress:
$HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" |


/usr/bin/gammu sendsms TEXT $CONTACTADDRESS1$
  }

  # 'notify-service-by-sms' command definition
  define command{
    command_name notify-service-by-sms
   command_line /usr/bin/printf "%b" "** Nagios **\n\nNotification Type:
$NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress:
$HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time:
$LONGDATETIME$\n\nAdditional Info:\n\n$SERVICEOUTPUT$" | /usr/bin/gammu
sendsms TEXT $CONTACTADDRESS1$
  }
```

Рисунок 2.5 - Налаштування конфігураційного файлу

Далі необхідно створити контакт або модифікувати існуючий.

Для роботи системи Nagvis потрібно, щоб на сервері були встановлені також:

• MySQL ;

• Nagios ;

• Apache [4] або інший web-сервер;

• Python з підтримкою бібліотек: apache2, gd, mysql, session, xml.

Для установки необхідно виконати наступні команди:

```
echo "dev-lang/php apache2 gd mysql session xml" >>
/etc/portage/packege.use
echo "net-analyzer/nagvis automap" >> /etc/portage/packege.use
emerge php mysql ndoutils nagvis
```

Крім перерахованих пакетів - програм, менеджер пакетів може запропонувати додаткові пакети необхідні для роботи Nagvis. Для роботи системи Nagvis, необхідно спочатку налаштувати пакет NDOUtils.

Для роботи NDOUtils необхідно створити базу даний в MySQL [17], в якій зберігатимуться дані моніторингу від системи Nagios .

Для створення бази даних потрібно виконати наступні команди:

```
mysql -u <user> -h <host> -p
mysq> create database nagios;
```

Змінна output_type може приймати три значення: file, tcpsocket і unixsocket. Відповідно якщо адміністратор встановлює тип tpcsocket, то він повинен вказати IP адреса і номер tcp порту. Якщо встановлено значення file, то змінної output також слід передати значення у вигляді повного шляху до файлу ndo.dat

Для того, щоб Nagios бачив NDOUtilits в конфігураційний файл необхідно додати наступні дані:

```
broker_module=/usr/bin/ndomod-3x.o config_file=/etc/nagios/ndomod.cfg
```

Для перевірки потрібно перезавантажити Nagios і відстежити, з’явилося чи в лозі повідомлення, що підтверджують, що все працює правильно:

```
/etc/init.d/nagios restart

tail -f /var/nagios/nagios.log
...
ndomod: NDOMOD 1.4b9 (10-27-2009) Copyright (c) 2009 Nagios Core
Development Team and Community Contributors
ndomod: Successfully connected to data sink.  0 queued items to flush.
Event broker module '/usr/local/nagios/bin/ndomod.o' initialized
successfully.
...
```

Базова конфігурація Nagvis знаходиться у файлі nagvis.ini.py. У ньому для базового налаштування необхідно вказати наступні дані:

```
[global]
language="english"
refreshtime=60

[defaults]
backend="ndomy_1"
icons="std_medium"
recognizeservices=1
onlyhardstates=0
usegdlibs=1
backgroundcolor="#fff"
headertemplate="default"
hovertemplate="default"
hoverdelay=0
showinlists=1
urltarget="_self"
```

У подальшому ці дані можна буде відредагувати через web-інтерфейс. Для того, щоб до Nagvis можна було звертатися з меню Nagios, потрібно додати посилання в файл /usr/share/nagios. Створити карти також можна з інтерфейсу системи Nagvis .

Для установки PNP доповнення до системи Nagios, потрібно виконати команду: emerge pnp4nagios. Далі треба зробити посилання в каталог /usr/share/nagios/htdocs, щоб легше було інтегрувати в Nagios .

ln - s/usr/share/pnp/usr/share/nagios/htdocs/pnp

Основні конфігураційні файли знаходяться в папці /etc/pnp/. Деякі конфігураційні файли можуть не використовуватися. Зокрема нам необхідний файл process_perfdata.cfg, тому потрібно виконати команду:

```
cp /etc/pnp/process_perfdata.cfg-sample /etc/pnp/process_perfdata.cfg
```

Далі необхідно відредагувати цей конфігураційний файл. Це можна зробити командою: nano/etc/pnp/process_perfdata.cfg

Як було описано раніше, є три способи для інтеграції pnp і Nagios. На думку дипломанта, необхідно використовувати метод, що забезпечує найбільшу продуктивність системи. У цьому методі обробку даних, для візуалізації у вигляді графіків виконує не Nagios, а служба NPCD .

Для налаштування методу «Bulk Mode with NPCD» потрібно спочатку настроїти метод «Bulk Mode». Для цього треба відкрити на редагування файл nagios.cfg командою: nano/etc/nagios/nagios.cfg і змінити рядки:

```
process_performance_data=1
service_perfdata_file=/var/nagios/service-perfdata
service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\tTIMET::
$TIMET$\tHOSTNAME::$HOSTNAME$\tSERVICEDESC::
$SERVICEDESC$\tSERVICEPERFDATA::$SERVICEPERFDATA$\tSERVICECHECKCOMMAND::
$SERVICECHECKCOMMAND$\tHOSTSTATE::$HOSTSTATE$\tHOSTSTATETYPE::
$HOSTSTATETYPE$\tSERVICESTATE::$SERVICESTATE$\tSERVICESTATETYPE::
$SERVICESTATETYPE$
service_perfdata_file_mode=a
service_perfdata_file_processing_interval=15
service_perfdata_file_processing_command=process-service-perfdata-file


host_perfdata_file=/var/nagios/host-perfdata
host_perfdata_file_template=DATATYPE::HOSTPERFDATA\tTIMET::
$TIMET$\tHOSTNAME::$HOSTNAME$\tHOSTPERFDATA::
$HOSTPERFDATA$\tHOSTCHECKCOMMAND::$HOSTCHECKCOMMAND$\tHOSTSTATE::
$HOSTSTATE$\tHOSTSTATETYPE::$HOSTSTATETYPE$
host_perfdata_file_mode=a
host_perfdata_file_processing_interval=15
host_perfdata_file_processing_command=process-host-perfdata-file
```

Потім, щоб використовувати метод «Bulk Mode with NPCD» потрібно змінити дві команди в конфігураційному файлі /etc/nagios/commands.cfg.

Для того щоб можна було використовувати служби NPCD, потрібно активувати конфігураційний файл /etc/pnp/npcd.cfg.

І перевірити шлях в рядках:

```
perfdata_spool_dir = /var/spool/pnp
perfdata_file_run_cmd = /usr/libexec/process_perfdata.pl
```

Далі потрібно запустити npcd службу і додати її в автозапуск:/etc/init.d/npcd start

Щоб до pnp було посилання з меню Nagios потрібно додати в /usr/share/nagios/htdocs/side.php наступний рядок:

```
<a href="pnp/" target="<?php echo $link_target;?>">PNP4Nagios</a>
```

Щоб графіки можна було спостерігати з розділу стану сервісів , потрібно створити файл /usr/share/nagios/htdocs/ssi/status - header.ssi і додати в нього наступні рядки:

```
<script src=\"/pnp4nagios/media/js/jquery-min.js\"
type=\"text/javascript\"></script>
<script src=\"/pnp4nagios/media/js/jquery.cluetip.js\"
type="text/javascript\"></script>
<script type=\"text/javascript\">
$(document).ready(function() {
  $('a.tips').cluetip({ajaxCache: false, dropShadow: false,showTitle: false
});
});
</script>
```

Тепер можна перезавантажити Nagios: /etc/init.d/nagios restart

Для використання розробленого компонента по збору даних, необхідно на сервері моніторингу встановити частини компонента-менеджера. Даний менеджер буде приймати дані від компонента-агента і передавати їх системі моніторингу.

Для установки компонента необхідно створити директорію за структурою схожу з структурою системи Portage і додати шлях до директорії в основний конфігураційний файл /etc/make.conf:

```
echo 'PORTDIR_OVERLAY="/usr/local/overlay"' >> /etc/make.conf
```

Далі створюємо папку для компоненти:

```
mkdir -p /usr/local/overlay/net-analyzer/hdtp/
```

Далі необхідно завантажити ebuild скрипт і додати його в систему. Далі запускаємо установку, система може запропонувати встановити додаткові пакети, які також необхідні: emerge hdtp. У процесі установки компонента в менеджері завдань створюється завдання перевірки нових даних.

2.2 Тестування

Бібліотекою setuptools передбачена можливість тестування пакету до його встановлення, за допомогою виконання всередині пакету команди

python setup.py test

Це можливо, якщо в функцію setup переданий аргумент test_suite. Даний аргумент повинен вказувати на функцію або інший виконуваний об’єкт, який здійснює запуск тестів для даного пакету.

Іншими корисними опціями для організації тестування є tests_require і test_loader. За допомогою tests_require можна передати список пакетів необхідних для запуску тестів. Даний аргумент схожий на install_requires, але пакети, зазначені в ньому, якщо вони відсутні в системі, не встановлюються. Перед запуском тестів вони будуть автоматично завантажені і розпаковані в папку з додатком. До їх залежностей застосовується той же сценарій. Це корисно, якщо для тестування використовуються додаткові бібліотеки, які не потрібні для роботи програми.

За допомогою test_loader можна вказати для пошуку тестів свій алгоритм. Він повинен міститися в класі з методом loadTestsFromNames().

Можливі два варіанти вирішення даної проблеми:

1. Пакет додатку крім самого додатку містить проект, налаштований для тестування цього додатку. Виконуваний об’єкт, зазначений в test_suite, запускає тести, використовуючи налаштування цього проекту.

```
#This file mainly exists to allow python setup.py test to work.
import os, sys
os.environ['SETTINGS_MODULE'] = 'test_project.settings'
test_dir = os.path.dirname(__file__)
```

```
sys.path.insert(0, test_dir)
from server.test.utils import get_runner
from server.conf import settings
def runtests():
   test_runner = get_runner(settings)
   failures = test_runner([], verbosity=1, interactive=True)
   sys.exit(failures)
if __name__ == '__main__':
 runtests()
```

Якщо додаток дуже простий і немає необхідності то всі необхідні настройки проекту можна вказати в скрипті для запуску тестів (рисунок 4.1):

```
#!/usr/bin/env python
from server.conf import settings
from server.core.management import call_command
APP_NAME = 'my_app'
settings.configure(
        INSTALLED_APPS=(
                'server.contrib.contenttypes',
                'server.contrib.sites',
                APP_NAME,
        ),
        # Server replaces all of this, but it still wants it. *shrugs*
        DATABASE_ENGINE='site',
        DATABASE_NAME=':memory:',
        ROOT_URLCONF='%s.urls' % APP_NAME,
        USE_L10N=True,
        DATE_FORMAT="d.m.Y",
        DATETIME_FORMAT="d.m.Y  H:i",
        TIME_FORMAT="H:i",
        SITE_ID=1,
)

def runtests():
        import server.test.utils
        runner_class = server.test.utils.get_runner(settings)
```

```
        test_runner = runner_class(verbosity=1, interactive=True)
        failures = test_runner.run_tests([APP_NAME])
        sys.exit(failures)
if __name__ == '__main__':
        runtests()
```

Важливим моментом в даному прикладі є імпорт server.test.utils. Налаштування Server-проекту повинні бути встановлені до його імпорту, інакше отримаємо помилку, що Server не може знайти файл з настройками проекту.

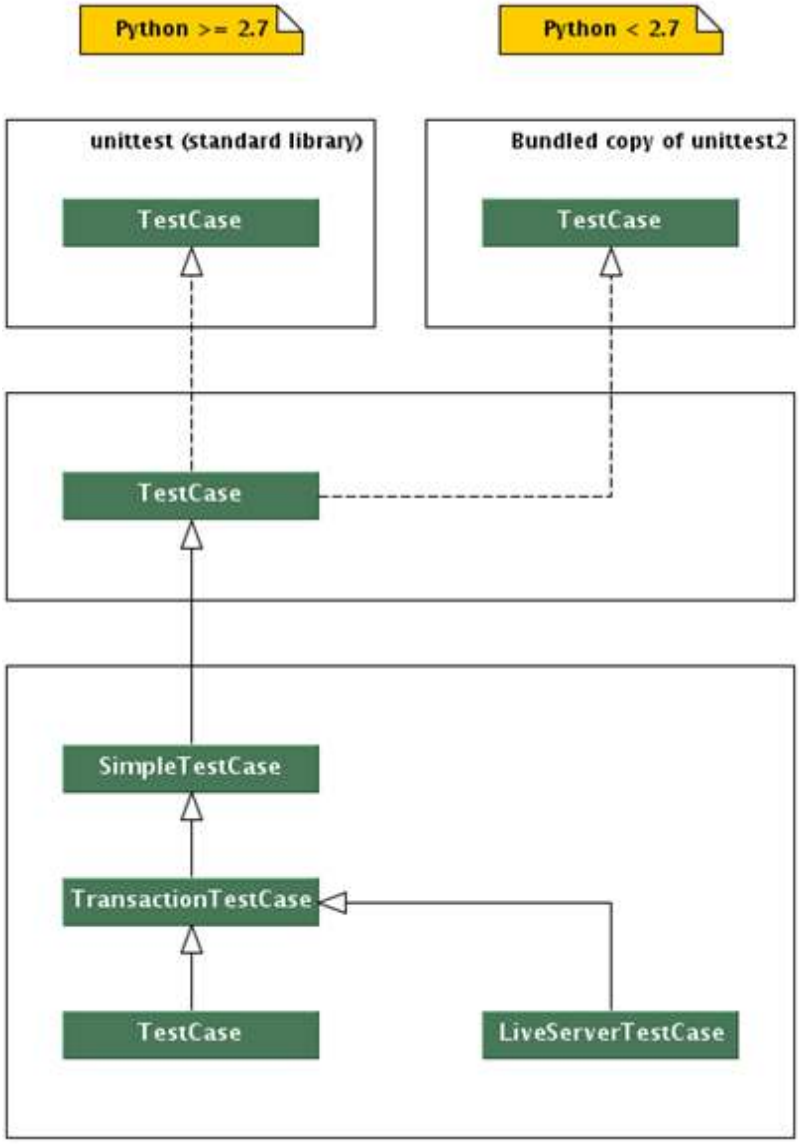

Рисунок 2.6 - Ієрархія класів модульного тестування проекту

Припустимо нам потрібна гарантія, що додаток буде стабільно працювати з різними версіями інтерпрітатора python. Ми можемо встановити необхідні версії в систему і для тестування нашого додатку розгорнути кілька віртуальних середовищ за допомогою virtualenv. У процесі розробки додатку перед запуском тестів необхідно в кожне віртуальне середовище розгорнути нову версію програми і запустити тести окремо для кожного середовища. Автоматизувати даний процес можна за допомогою tox. Звичайно набагато крутіше для цього використовувати Jenkins або іншу систему безперервної інтеграції, але для розробника від tox результат тестування буде отримано швидше (зазвичай щоб код потрапив на сервер інтеграції, необхідно спочатку додати його в репозиторій).

Для тестування програми за допомогою tox необхідно до його дистрибутива додати файл tox.ini, що містить налаштування для розгортання віртуальних середовищ. Для кожного середовища можна вказати набір залежностей і команди, які будуть виконані в процесі тестування.

Приклад конфігурації і побудовою звіту по покриттю коду тестами (звіт будується тільки для одного середовища, тести виконуються для обох):

```
[tox]
downloadcache = .tox/_download/
envlist = py26, py27
[testenv]
deps =
        mock
commands =
        {envpython} setup.py test
[testenv:py27]
deps =
        coverage
        mock
commands =
   coverage run --branch --source=news_line setup.py test
   coverage report --omit=news_line/tests.py,news_line/migrations/*
```

coverage html --omit=news_line/tests.py,news_line/migrations/ -d reports/coverage/*

Запуск тестів проводиться за допомогою команди tox. Що відбувається після запуску:

- tox створює дистрибутив програми;
- розгортає, зазначені в конфіги віртуальні середовища;
- встановлює дистрибутив програми;
- запускає тести для дистрибутива (команди, вказані в конфігах).

Виконання кожного етапу супроводжується виведенням на екран і в файли журналів.

Для здійснення процедури функціонально-інтеграційного тестування необхідно спочатку встановити необхідні для цього пакети:

$ pip install coverage >= 3.0

$ pip install webtest

$ pip install server-webtest

$ pip install server-coverage

WebTest - це бібліотека для функціонального тестування WSGI-додатків. WebTest більш потужний API, функціональні тести за його допомогою писати простіше. WebTest буде працювати набагато швидше, мати кращу інтеграцію з іншим кодом і більш простими налаштуваннями.

Налаштовуємо проект. Невелика настройка буде потрібно для server-coverage:

додати 'server_coverage' в INSTALLED_APPS і в settings.py вказати, куди зберігати html-звіти.

COVERAGE_REPORT_HTML_OUTPUT_DIR = os.path.join (PROJECT_PATH, 'cover')

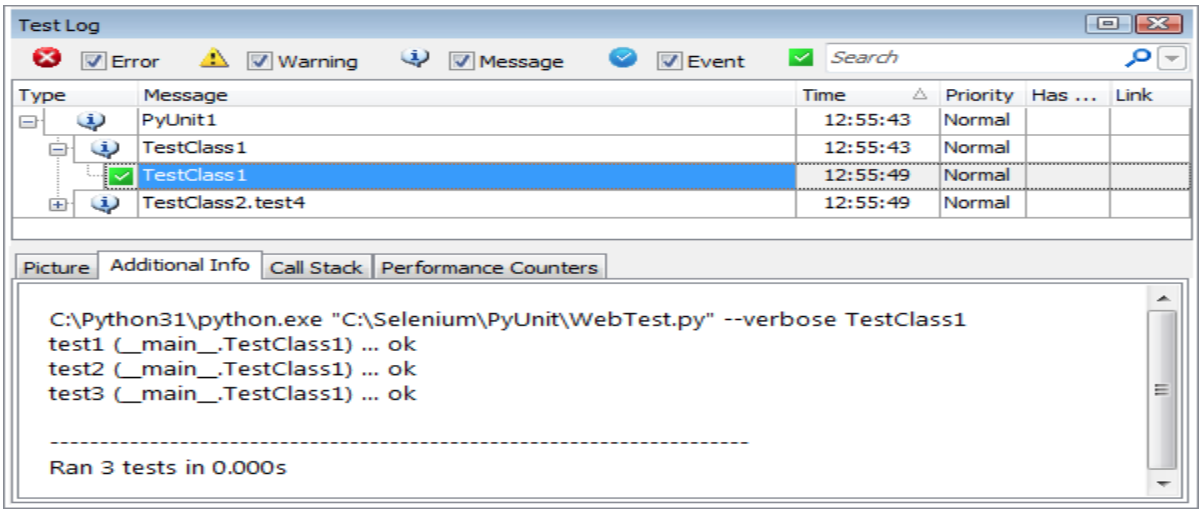

Рисунок 2.7 - Тестування Python web-додатку

Розглянемо функціональний тест для реєстрації та авторизації в системі:

Зберігаємо його в файлі tests.py нашого додатку в проекті. Перевіряємо, чи може зареєстрована людина вийти з сайту, потім зайти на нього (ввівши свої email і пароль), чи може зареєструватися (лист отримано, посилання на активацію вірне), чи потрапляє на потрібну сторінку після активації облікового запису. Для зручності використовувалася також фікстура, в якій вже підготовлений користувач vadym з email= vadym@gmail.com (і яка використовується і в інших тестах). Можна було цього користувача прямо в тесті створити, це не міняє суті.

При кожному переході WebTest автоматом перевіряє, що нам повернувся код 200 або 302 (це налаштовується). Для відправки форм не потрібно конструювати POST-запити вручну, форми підхоплюються з html-коду відповіді, досить присвоїти значення за потрібних полях і виконати метод submit(). Переходи по редіректах після POST-запитів робляться руками (і це корисно, тому що якщо редиректу немає - наприклад, помилка при заповненні форми, то тест це покаже).

server_webtest.WebTest - це нащадок від TestCase, вміє все те ж. Але головне - в ньому доступна змінна self.app типу ServerTestApp (це типу webtest.TestApp), через яку можна отримати доступ до API WebTest. Якщо user

переданий, то запит (ну і всі наступні переходи по посиланнях, відправки форм) буде виконаний від імені джанговского користувача з цим username'ом.

Створюємо файл test_settings.py (в корені проекту) приблизно такого змісту ():

*from settings import **

DATABASE_ENGINE = 'site_pos'

DATABASE_NAME = 'testdb.'site_pos'

А потім запускаємо тести:

$ Python manage.py test_coverage myapp1 myapp2 myapp3 --settings = test_settings

Можна і без test_settings обійтися (запускати просто $python manage.py test_coverage myapp і ніяких додаткових файлів не створювати), просто з ним зручніше: можна туди будь-які специфічні для тестів настройки написати, наприклад, використовувати іншу СУБД для більш швидкого виконання тестів або підміняти URLOpener для urllib2, щоб тести не лізли в інтернет. Команду для запуску тестів зручно обернути в shell-скрипт (або bat-файл).

Звіт по code coverage зберігся у зазначеній раніше папці. Відкриваємо його (файл cover / index.html) і бачимо щось на зразок цього, що представлено на рисунку 2.8:

Test Coverage Report

Generated: 2019-04-19 23:52 YEKST

Module	Statements			
	total	executed	excluded	% covered
accounts.admin	15	11	4	100.0%
accounts.backends	44	8	7	21.6%
accounts.forms	59	49	10	100.0%
accounts.middleware	7	7	0	100.0%
accounts.models	43	30	13	100.0%
accounts.utils	8	2	0	25.0%
accounts.views	24	21	3	100.0%
cache_utils.group_backend	79	45	6	61.6%

Рисунок 2.8 - Результати тестування web-додатку

Переходимо за яким-небудь посиланням і бачимо, який код у нас виконався під час тестів, а який - не виконався (і, отже, ніяк не міг бути протестований):

friendly_auth.backends.email_backend: 52 total statements, 73.8% covered
Generated: Tue 04-20 04:48 YEKST
Source file: /home/nadovmeste/src/nadovmeste/friendly_auth/backends/email_backend.py
Stats: 31 executed, 11 missed, 10 excluded, 54 ignored

```
1. #coding: utf-8
2.
```

Рисунок 2.9 - Результати тестування web-додатку

```
81.
82. class EmailRegistrationForm(forms.Form):
83.
84.     email = forms.EmailField(widget=forms.TextInput(attrs=dict(attrs_dict,
85.                                                                maxlength=75)),
86.                              label=_("Email address"))
87.     password1 = forms.CharField(widget=forms.PasswordInput(
88.                                           attrs={'class': 'required',
89.                                                  'autocomplete':'off'},
90.                                           render_value=False),
91.                                 label=_("Password"))
92.
93.
94.     def clean_email(self):
95.         """
96.         Validate that the supplied email address is unique for the
97.         site.
98.
99.         """
100.        if User.objects.filter(email__iexact=self.cleaned_data['email']):
101.            raise forms.ValidationError(_("This email address is already in use. Please supply a different email address."))
102.        return self.cleaned_data['email']
103.
104.
```

Рисунок 2.10 - Результати тестування web-додатку

Функціональне тестування - це тестування програмного забезпечення з метою перевірки можливості реалізації функціональних вимог, тобто здатності програмного забезпечення в певних умовах вирішувати завдання, потрібні користувачам. Функціональні вимоги визначають, що саме робить програмне забезпечення, які завдання воно вирішує.

Набір тестів на функціональність представлений в таблиці 4.1. Перевірка ергономічності - метод оцінки зручності продукту у використанні, заснований на залученні користувачів в якості тестувальників, випробувачів і підсумовуванні отриманих від них висновків.

При проходженні даного тесту були залучені фахівці з позиціонування та просування сайтів компанії. Тестувальникам було запропоновано вирішити такі завдання:

- зареєструватися;

- переглянути сервери, моніторингом яких вони займаються;

- переглянути статус обладнання;

- проаналізувати активність основних показників;

- додати запити в створені проекти, грунтуючись на рекомендаціях для моніторингу;

- отримати звіт про стан моніторингу;

- видалити запити з проекту.

Таблиця 2.1

Набір функціональних тестів системи

№	Назва	Кроки	Очікуваний результат	Результат тестування
1	Реєстрація	1. На сторінці входу натиснути кнопку «Реєстрація». 2. Заповнити форму реєстрації. 3. Натиснути кнопку «Готово».	У базі даних сервера з'явився новий користувач.	Так
2	Вхід в систему	1. На сторінці аутентифікації ввести зареєстровану адресу електронної пошти і вірний пароль. 2. Натиснути на кнопку «Увійти».	Користувачеві стає доступна головна сторінка сайту	Так
3	Виявлення доступних серверів	1. На сторінці одного з проектів вибрати вкладку «Доступні сервери»	На сторінці відображається список доступних серверів	Так
4	Перегляд статусу обладнання	1. На сторінці одного з проектів вибрати вкладку «Сервери». 2. Вибрати відповідне обладнання	Поруч з вибраними сервером відображається список обладнання, яке використовується для моніторингу.	Так
5	Отримати звіт про стан моніторингу	1. На сторінці моніторингу. 2. Вибрати проміжок часу. 3. Натиснути на кнопку «Отримати звіт».	Дані зі сторінки зберігаються на комп'ютер користувача у вибраному форматі.	Так

Всі завдання були вирішені усіма тестувальниками без будь-яких ускладнень. Тест пройдено успішно. Далі проводилося тестування на працездатність. Тестування пройшло успішно.

Тестування інтерфейсу користувача. Інтерфейс був протестований на браузерах Internet Explorer, Mozilla Firefox, Opera, Google Chrome. Не виникло жодних проблем з версткою. Інтерфейс у всіх браузерах був ідентичним. Тест пройдено успішно.

Робота системи була протестована на 50 проектах, наявних в системі. У кожному проекті міститься близько 10 серверів. На всіх запитах були отримані коректні результати при тестуванні сервісу моніторингу і сервісу створення рекомендацій для підвищення його якості.

Для впровадження даного рішення і подальшого розгортання необхідно створити зручний для цього інструмент. Для виконання даного завдання необхідно проаналізувати часто використовувані інструменти в Linux системах.

Розглянемо наступні інструменти:

• файл ebuild, який використовується в операційній системі Gentoo Linux;

• rpm пакет, так як даний вид інсталяційного пакету дуже поширений в середовищі Linux;

• tar архів з інсталяційним скриптом, так як даний вид інсталяції пакетів дуже поширений для установки пакетів не залежних від дистрибутива Linux.

RPM (Red Hat Package Manager) - це відкрита система управління пакетами. Вона дозволяє формувати пакет для програм у вигляді вихідних кодів, з можливістю зручної побудови та встановлення, а також пакети для вже зібраних програм, з можливістю їх установки і видалення. Також дана система підтримує базу даних. Компанія Red Hat підтримує використання даного менеджера в різних дистрибутивах Linux. RPM дуже простий для виправлень та використання, а також може використовуватися як базовий менеджер пакетів у великих системах.

RPM дозволяє шукати програми в особистій базі даних, звіряти версії і цілісність пакетів, а також отримувати дані про програму і про пакети. Хоча самі дані в пакеті знаходяться в стисненому стані, але на вимогу вони швидко розпаковуються.

Для використання RPM пакетів необхідно встановити менеджер пакетів RPM. Це можна зробити командою: emerge rpm.

Основні команди для роботи з RPM пакетами. Для встановлення RPM пакету, необхідно виконати команду rpm з специфікатором - i:

rpm - i package_name.rpm (якщо файл знаходиться на ftp або web-сервері то можна вказати посиланням);

Для видалення пакета, необхідно виконати команду rpm з специфікатором e: rpm - e package_name.rpm.

Для виведення даних про rpm пакет, потрібно виконати команду rpm з специфікатором qpi: rpm - qpi package_name.rpm

У результаті виконання даної команди буде виведена вся інформація про пакет, наприклад:

```
Name        : koules                        Distribution: Red Hat Linux
Colgate
Version     : 1.2                              Vendor: Red Hat Software
Release     : 2                             Build Date: Mon Sep 02 11:59:12
1996
Install date: (none)                        Build Host: porky.redhat.com
Group       : Games                         Source RPM: koules-1.2-2.src.rpm
Size        : 614939
Summary     : SVGAlib action game with multiplayer, network, and sound
support
Description :
This arcade-style game is novel in conception and excellent in execution.
No shooting, no blood, no guts, no gore.  The play is simple, but you
still must develop skill to play.  This version uses SVGAlib to
run on a graphics console.
```

Рисунок 2.11 - Інформація про RPM пакет

Для створення RPM пакетів необхідно створити дерево каталогів, яке матиме певну структуру:

- BUILD - директорія в якій збираються вихідні коди;

- RPMS - директорія в яку поміщаються зібрані бінарні RPM пакети;

- SOURCES - директорія в якій знаходяться вихідні коди програм;

- SPECS - директорія в якій знаходяться файли з розширенням *. Spec, що представляють з себе інструкцію по збору пакетів;

- SRPMS - директорія в яку поміщаються RPM пакети з вихідним кодом.

Як правило, це структура створюється автоматично і розміщується в /usr/src/RPM, при установці пакета rpm.

Проаналізувавши даний метод установки пакетів можна зробити висновок, що використання rpm пакетів не є зручним способом для розгортання рішення або його компонентів. Це пов’язано з тим, що система Nagios має свій менеджер по установці пакетів, а для використання rpm пакетів необхідно встановлювати на всіх серверах додатковий менеджер пакетів.

Ebuild [15] скрипт є основним компонентом системи Portage. У свою чергу система Portage є основним джерелом програм в операційній системі Gentoo Linux. Ebuild скрипт містить в собі всю інформацію необхідну для завантаження, розпакування, складання і встановлення програм, як правило зберігаються у вихідних кодах, а також в даному скрипті знаходяться опції для кожного кроку установки і видалення програм. Велика частина системи Portage написана мовою Python, тоді як Ebuild скрипти написані на мові командного рядка bash, так як використання bash дозволяє нам викликати команди з командного рядка. Один з найважливіших принципів розробки ebuild скриптів, це використання команд аналогічних командному рядку, так якби знадобилося встановлювати пакет вручну.

Ebuild скрипти інтерпретуються командами ebuild і emerge, де команда ebuild представляє спрощений варіант програми для зборки. Вона дозволяє збирати і встановлювати тільки поодинокі ebuild скрипти. Вона також дозволяє побачити якщо є залежності, але не має можливості їх вирішення. З іншого боку команда emerge це потужний інструмент для роботи з ebuild скриптами, вона дозволяє вирішувати залежності, якщо це необхідно. Команда emerge дозволяє відстежувати кожен етап збірки пакету.

Python 3 вимагає настройки для підключення до MySQL. Існує багато варіантів настройки (наприклад, MySQLclient), але для простоти в цьому питанні використовується pymysql. Встановлюємо цей модуль за допомогою Pip:

sudo pip3 install pymysql

Вставновленн Apache. Далі потрібно встановити Apache і переконатися, що веб-сервер розпізнає файли Python як виконувані. Встановлюємо Apache за допомогою apt-get:

sudo apt-get install apache2

Як і MySQL, сервер Apache запуститься відразу після установки.

Тепер потрібно помістити root-каталог сайту в надійну точку системи. Стандартний каталог знаходиться в / var / www / html. Дотримуючись інструкції, створємо тестовий підкаталог на ім'я test в цьому каталозі.

sudo mkdir / var / www / test

На завершення потрібно налаштувати взаємодію Python і Apache. Відключаємо багатопотокові процеси.

sudo a2dismod mpm_event

Після цього надаємо Apache розширені права на запуск скриптів.

sudo a2enmod mpm_prefork cgi

Потім міняємо налаштування Apache, щоб явно оголосити файли Python як виконувані і дозволити запускати їх. Відкрийте конфігураційний файл за допомогою nano або будь-якого іншого текстового редактора.

sudo nano /etc/apache2/sites-enabled/000-default.conf

Після рядка <VirtualHost *: 80 \> додаємо наступний код:

<Directory / var / www / test>
Options + ExecCGI
DirectoryIndex index.py
</ Directory>
AddHandler cgi-script .py

Перевіряємо, що блок <Directory> поміщений в блок <VirtualHost>.

/etc/apache2/sites-enabled/000-default.conf
*<VirtualHost *: 80>*
<Directory / var / www / test>
Options + ExecCGI
DirectoryIndex index.py
</ Directory>
AddHandler cgi-script .py

Цей блок Directory дозволяє налаштувати поведінку Apache для цього каталогу. Він повідомляє Apache, що каталог / var / www / test містить виконувані файли, задає index.py в якості стандартного файлу і потім визначає виконувані файли.

Також потрібно дозволити виконувані файли в каталозі сайту. Для цього змінюємо шлях для DocumentRoot. Знаходимо стрічку, яка починається з DocumentRoot / var / www / html, і змінюємо її:

Зберігаємо і закриваємо файл. Щоб зміни вступили в силу, перезавантажуємо Apache.

sudo service apache2 restart

Тестування налаштування. Тепер потрібно перевірити, чи працює зв'язка належним чином. Для цього створюємо тестову веб-сторінку і БД.

Для початку створіть БД. Увійдіть в MySQL, ввівши root-пароль.

mysql -u root -p

Тепер створюємо тестову БД (для прикладу назвемо її example):

CREATE DATABASE example;

Відкриваємо нову БД:

USE example;

Додаємо таблицю для даних Python:

CREATE TABLE numbers (num INT, word VARCHAR (20));

Натискаємо CTRL + D, щоб вийти.

Потім створюємо новий файл для простого додатку Python.

sudo nano /var/www/test/index.py

Зберігаємо і закриваємо файл. Потім встановлюємо права на новий файл.

sudo chmod 755 /var/www/test/index.py

За допомогою браузера відкриваємо http://ip_адрес_сервера. З'явиться наступне інформаційне повідомлення:

http: // your_server_ip

[(1, 'One!'), (2, 'Two!'), (3, 'Three!')]

Тепер сервер підтримує Python 3 з надійною базою даних на бекенді. Крім того, управління пакетами стало набагато простіше, оскільки на сервері встановлено зручні менеджери пакетів.

Однак на даному етапі сервер стає вразливим і потребує додаткового налаштування. SSL-шифрування не є обов'язковим компонентом для роботи сервера, однак це може значно підвищити рівень безпеки.

2.3. Інструкція користувача

З огляду на всі вищеописані вимоги до моніторингу, було вирішено використовувати систему, яка могла б працювати з будь-якими операційними системами, могла б виконувати скриптові сценарії на більшості поширених мов і не вимагала б установки агентського програмного забезпечення на кожну машину, за станом якої буде здійснюватися моніторинг.

Було відмічено, що система Nagios, яка вже використовується в інфраструктурі проекту, (за допомогою неї виконувалося розгортання проекту на сервера), повністю відповідає необхідним умовам, а також не вимагає вкладення додаткових матеріальних засобів і установки пакетів сервера і агента.

У загальному випадку моніторинг в Nagios є окремим проектом (проектами) з однією або декількома конфігураціями усередині: розбиття на конфігурації краще здійснювати в залежності від необхідності сповіщень тих чи інших користувачів про певні проблеми, тобто оповіщення спрацьовує при будь-якій помилці всередині конфігурації, яка призводить до невдалої побудови.

Також можна не створювати окремий проект, а додати конфігурації (конфігурацію) до існуючого проекту розробки, на серверах якого реалізовується моніторинг.

Перше зручно в разі великої компанії, коли розробкою і налагодженням поломок сервера займаються різні люди, другий випадок можна

використовувати в маленькій компанії (в невеликому проекті), коли всі маніпуляції здійснюються силами розробників. Вид проекту представлений на рисунку 2.12

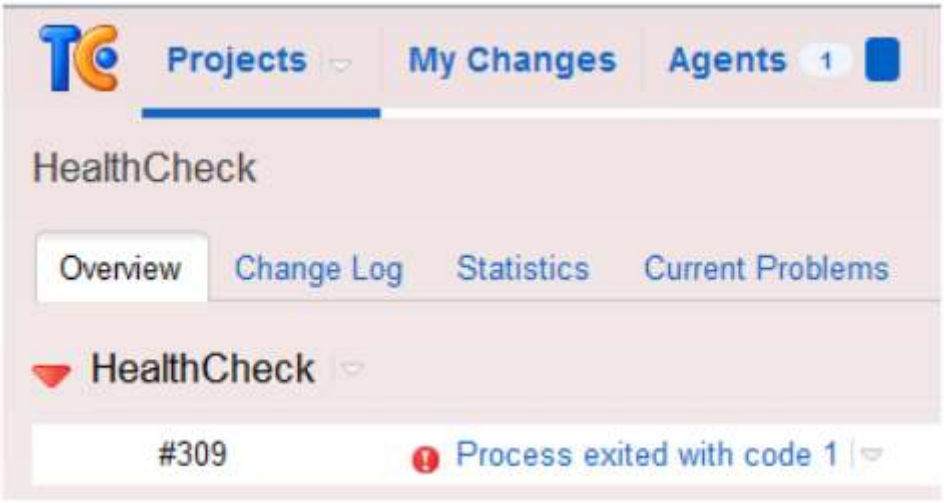

Рисунок 2.12 - Загальний вид проекту

У проекті містяться конфігурації для виконання, кожна з яких може складатися з одного або декількох кроків (рисунок 4.8). Крок в даному випадку являє собою скрипт (найчастіше з параметрами). При виконанні кроку перевіряється значення відслідковуваного показника.

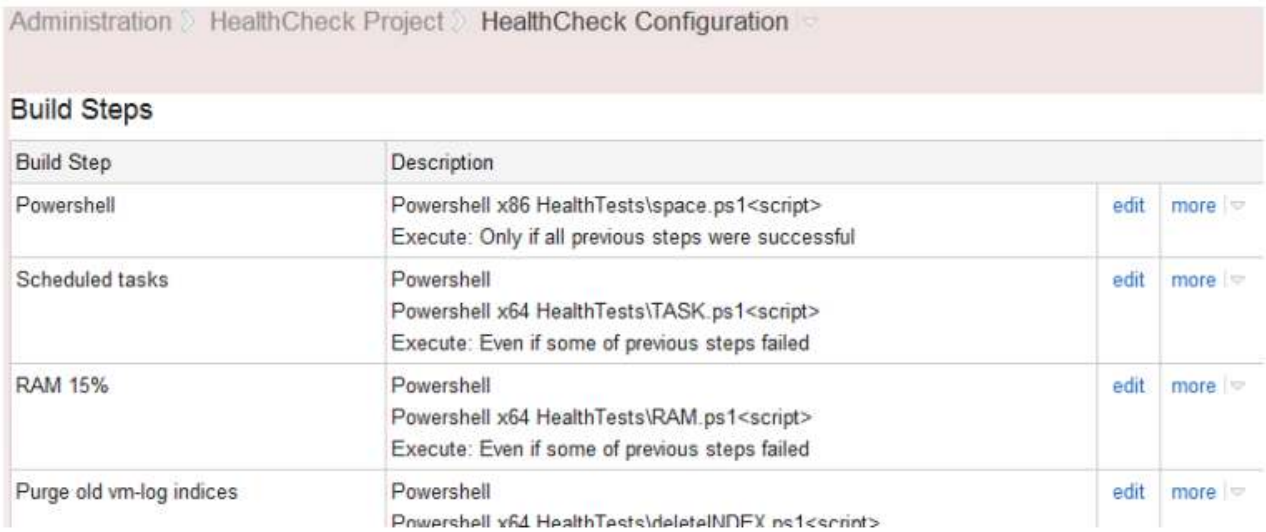

Рисунок 2.13 - Покрокова стратегія конфігурації

Всі дії логуються (всередині також пишуться призначені для користувача

повідомлення, які описані всередині скрипта). Стан виконаних конфігурацій зберігаються в історії проекту, де завжди можна подивитися детальний лог виконання (рисунок 2.14).

Рисунок 2.14 - сторія збірки конфігурації

Вся інформація з історії збірок зберігається в базі даних (в нашому випадку це MySQL), і може бути використана аналітичним відділом для обробки даних про доступності інфраструктури протягом певного часу.

При збої користувачі, які налаштували для себе оповіщення даного проекту або конфігурації отримують повідомлення, яке представлено на рисунку 2.15 або будь-якого іншого (налаштовані шаблони).

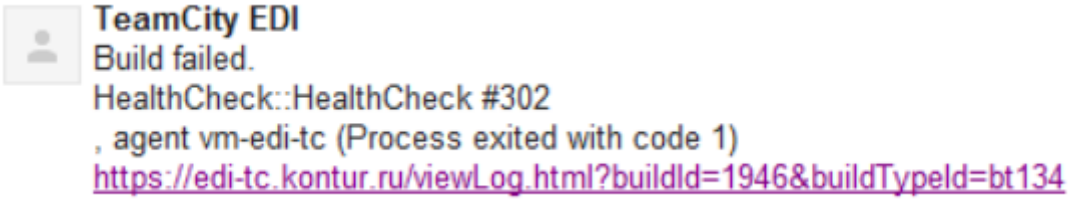

Рисунок 2.15 - Повідомлення про збій

В даному шаблоні міститься посилання на докладний лог конфігурації, при виконанні якої сталася помилка. Збірка конфігурації здійснюється за заданим адміністратором розкладом: визначення розкладу може відбуватися як

в стандартній формі, прийнятій, наприклад, в планувальнику завдань, так і у вигляді стандартного cron виразу.

Тригерів складання може бути задано необмежену кількість. Також можливо накласти різні умови, при яких той чи інший тригер буде чи не буде активним у певний момент часу. Загальний вигляд сконфігурованого тригера представлений на рисунку 2.16.

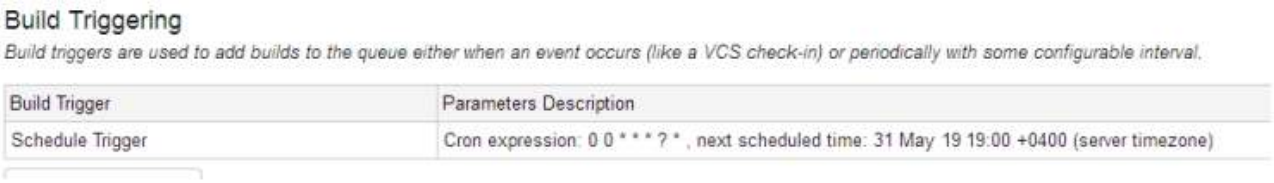

Рисунок 2.16 - Розклад збірки

Крім того, можна використовувати вбудовані засоби для моніторингу стану, збору статистики, побудови звітів. Кінцевий вигляд статистики можна спостерігати на рисунку 2.17.

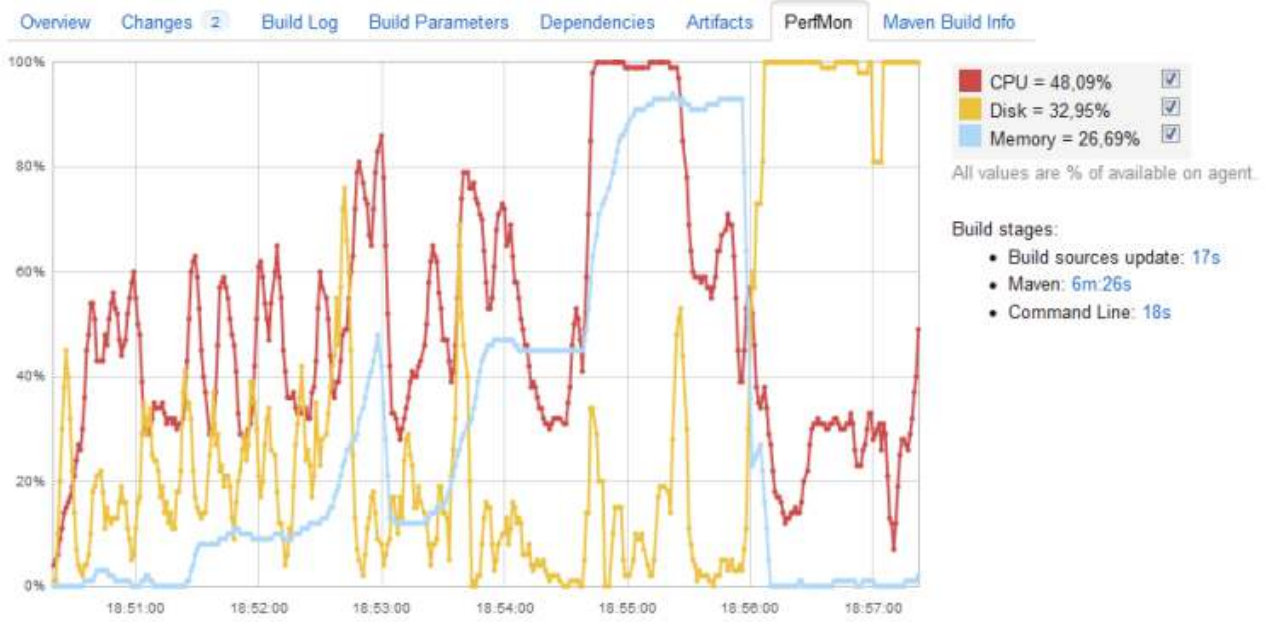

Рисунок 2.17 - Статистика

Дуже зручна можливість - є клікнути в точку на графіку – буде виведено лог, який відповідає цьому моменту часу - таким чином, можна подивитися що саме створювало навантаження на BuildAgent.

Збірка конфігурації повинна здійснюватися на машині з встановленим агентськими програмним забезпеченням, проте звернення до всіх машин можна здійснювати з єдиного розгорнутого агента, тобто встановлення на всі сервера не потрібно.

Однак за рахунок налаштування декількох агентів можна здійснити масштабування моніторингу. Розподіл збірок здійснюється за рахунок спеціального параметра збірки - вимоги до агента (рисунок 2.18).

Існує стандартний набір параметрів, за яким можна розподілити агенти на пули, але також можна задавати і свої власні умови поділу. Крім вимог до агентів, можна використовувати також параметри збірки.

Рисунок 2.18 - Вимоги до агента

Зазвичай туди поміщаються спеціалізовані змінні. Наприклад, аутентифікаційні дані (існує спеціальний вид параметрів, які шифруються після створення, що дозволяє зберігати паролі в закритому вигляді і використовувати їх при виконанні конфігурації). Також в моніторингу проекту параметри збірки вказуються для скасування моніторингу сервера (наприклад, сервер відключений в зв’язку з ремонтними роботами і ви не хочете отримувати постійні оповіщення про його недоступності). Параметри можуть застосовуватися як для конкретної збірки, так і для проекту в цілому.

Скрипти для збірки не обов’язково зберігати прямо на агента. Для чекаута скриптів і змін в систему вбудованf підтримка різних VCS. В своєму проекті моніторингу я використовую Git. До проекту може бути прив’язане кілька систем контролю версій одночасно. Також можна вказати вид чекаута:

- чекаут на сервері, якщо ви не хочете навантажувати агент при складанні, або чекаут на агента, якщо ви вважаєте сам сервер слабким місцем у вашій системі.

У підсумку ми отримуємо багатогранну систему з широкими можливостями інтеграції, настройки і масштабування. Однак існує єдиний мінус - моніторингові скрипти все ще залишаються на совісті системного адміністратора.

Отже, застосування подібної системи для моніторингу серверів і служб проекту дозволить:

- підвищити експлуатаційну готовність серверної площадки;

- отримати оперативну інформацію про стан систем (також контролювати значення критично важливих метрик);

- швидше шукати першопричину збою і навіть прогнозувати його заздалегідь;

- контролювати конфігурацію обладнання;

- надавати інформацію для клієнтів (в тому числі для аналітичного відділу), на основі якої можливо зробити аналіз ступеня негативного впливу збою на послуги проекту.

Висновки до розділу 2

Здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. Розкрито питання встановлення та налаштування програмного забезпечення на сервері, а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

РОЗДІЛ 3
БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1. Режим роботи що забезпечують високу працездатність в галузі інформаційних технологій

Працюючи за комп'ютером, рекомендується дотримуватися правил тривалості роботи, правильної постави, розміру шрифтів та зображень, вимог до приміщення тощо. Принципи правильної роботи за комп'ютером:

- у робочому приміщенні (кімнаті), де встановлені комп'ютери, щодня потрібно виконувати вологе прибирання;
- приміщення, у якому знаходяться комп'ютери, потрібно провітрювати щогодини;
- після кожного часу роботи рекомендується робити десяти хвилинну перерву, яку зручно суміщати з провітрюванням. За будь-яких умов безперервна робота за комп'ютером для дорослої людини не повинна перевищувати двох годин. Під час перерви не варто читати або дивитися телевізор. Перерва, яку Ви проводите за комп'ютером (наприклад, граючись або шукаючи матеріали в Інтернеті), просто не має сенсу;
- необхідно постійно слідкувати за станом екрану монітора: він має бути чистим, без плям та пилу. Крім того, обов'язково слідкуйте за чистотою окулярів – комп'ютерних чи звичайних;
- слідкуйте за поставою: ноги твердо стоять на підлозі чи на спеціальній підставці; стегна розташовані під прямим кутом до тулуба, а гомілки – під прямим кутом до стегон; сидіти потрібно прямо або злегка нахилившись вперед; пальці рук знаходяться на рівні зап'ястків або трохи нижче – у такому положенні вони найбільш рухливі; плечі мають бути розслаблені та вільно опущені, що сприяє розслабленню рук; відстань від очей до екрану монітора – не менше 55-60 см; центр екрану має знаходитися на рівні

очей чи трохи нижче; рекомендується хоча б раз на день виконувати гімнастику для очей;

- щоб попередити „синдром сухого ока”, моргайте кожні 3-5 секунд;
- як не дивно, але й у наш час є люди, які замість монітору використовують звичайний телевізор. Так чинити категорично не рекомендується: випромінювання від телевізора практично у сто разів перевищує випромінювання монітора. Це зумовлено тим, що телевізор призначений для перегляду на значній відстані;
- у процесі роботи за комп’ютером обов’язково звертайте увагу на дихання: воно має бути рівномірним, без затримок;
- при роботі з текстом рекомендується, щоб колір шрифту був темним, а колір фону – світлим (ідеальний варіант – чорний шрифт на білому фоні);
- якщо шрифт занадто мілкий, то потрібно збільшити масштаб документу (наприклад, до 150% чи більше);
- при наборі текстів з паперів чи книг рекомендується помістити джерело якомога ближче до монітору. Це дозволить уникнути частих рухів головою та очима;
- якщо є можливість, міняйте вид діяльності, якою займаєтеся протягом дня;
- у процесі роботи рекомендується періодично (приблизно раз на 20-30 хвилин) переводити погляд з екрану на найбільш віддалений предмет у кімнаті, а ще краще – на віддалений об’єкт за вікном;
- якщо з’явилося відчуття втоми, напруження, сонливості, тяжкості в очах, потрібно припинити роботу та хоча б трохи відпочити.

Законодавчо встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ЕОМ при 8-годинній денній робочій зміні залежно від характеру праці:

- для розробників програм слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за персональним комп’ютером;
- для операторів персональних комп’ютерів слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;
- для операторів комп’ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за персональним комп’ютером.

3.2. Інженерний захист персоналу об’єкту та населення. Правила застосування.

Головною метою захисту населення і територій під час НС є забезпечення реалізації державної політики у сфері запобігання і реагування на НС та ліквідації їх наслідків. Заходи з підготовки до захисту населення проводяться завчасно по територіально-виробничому принципу створення системи цивільного захисту цивільної оборони.

При цьому вони ведуться як у зв’язку з можливими надзвичайними ситуаціями природного, техногенного та терористичного характеру, так і в передбаченні небезпек, які виникають при веденні воєнних дій, оскільки значна частина цих заходів ефективна, як у мирний час та і у воєнний період.

Кожний громадяни України зобов‘язаний дотримуватись заходів безпеки в побуті та повсякденній трудовій діяльності, не допускати порушень виробничої та технологічної дисципліни, вимог екологічної безпеки, охорони праці, що можуть призвести до надзвичайної ситуації, вивчати основні способи захисту населення і територій від наслідків НС, надання першої медичної допомоги постраждалим, правила користування засобами захисту.

Інженерний захист населення і територій, це комплекс інженерно-технічних заходів, який проводиться завчасно та в оперативному порядку, направлений на попередження або максимальне зниження втрат населення та матеріальних збитків при виникненні НС техногенного, природного, соціально-політичного та військового характеру.

Інженерний захист населення і територій займає важливе місце серед усього комплексу заходів захисту населення і територій, які виконуються посадовими особами і органами управління усіх рівнів.

Організація та проведення заходів інженерного захисту населення і територій являється обов'язковою функцією для центральних органів виконавчої влади, місцевих державних адміністрацій та органів місцевого самоврядування, підприємств і організацій, незалежно від їх організаційно-правових форм і форм власності.

З метою захисту населення і територій від НС, запобігання їх виникненню та зменшенню втрат і шкоди економіці держави, ефективної ліквідації наслідків НС проводиться спеціальний комплекс заходів захисту.

Для забезпечення захисту населення міст, працюючих і службовців об'єктів економіки, створюється ФЗС, проводиться інформування та оповіщення, плануються евакуаційні, медичні заходи, населення забезпечується засобами індивідуального захисту та приладами дозиметричного й хімічного контролю, проводиться підготовка населення до дій у НС. Порядок здійснення основних заходів у сфері захисту населення і територій від НС, а саме таких елементів як інформування та оповіщення, спостереження, укриття в захисних спорудах, евакуаційні заходи, хімічний і медичний захист та інші, визначається законами України «Про захист населення і територій від НС техногенного та природного характеру», «Про правові засади цивільного захисту», «Про зв'язок», «Про телебачення та радіомовлення, та інженерно-технічними заходами цивільного захисту».

ВИСНОВКИ

Бізнес в будь-якій сфері сильно зав’язаний на доступності і працездатності його IT-інфраструктури 24/7/365. Щоб забезпечити цю працездатність, необхідно заздалегідь виявляти вузькі місця в конфігурації систем і мереж, а також швидко дізнаватися про наявність поломки і її причини. Для цих потреб в компаніях, де подібне стеження нездійсненно за рахунок тільки фахівців, прийнято використовувати системи моніторингу.

Системи моніторингу бувають платні та вільно поширювані, а також розрізняються за своїм наповненням «з коробки», масштабованості, необхідних ресурсів і рівнем знань, необхідним для прийнятної настройки.

При виборі, розробці, впровадженні систем моніторингу спочатку потрібно визначитися з об’єктами, які будуть піддаватися стеженню, а також критичні події та показники, які і визначать кількість повідомлень при поломці, частоту сканування і інші параметри і наслідки. Причому оцінювання показників в першу чергу потрібно здійснювати не з точки зору технічного інженера, а з точки зору кінцевого користувача.

Хід виконання роботи повністю відповідав поставленим завданням. Для початку було висунуто загальні вимоги до моніторингу підтримуваних проектів, володіючи списком яких можна було приступити до пошуку відповідного готового рішення.

Потім докладно розглянуті різні існуючі системи моніторингу серверів від широкого різноманіття фірм-розробників, проаналізовані їх позитивні і негативні сторони, відповідність їх сформульованим раніше вимогам.

У підсумку, зроблено висновок про неможливість або неоптимальність використання відомих систем. Після чого критерії відповідності моніторингу поставленим завданням в проекті були максимально уточнені, завдяки чому став можливий вибір оптимальної основи для його створення - вже використовуваного для інших потреб розробки пакету Nagios.

У міру виникнення проблем і нових завдань, в проекті розроблялися різноманітні модулі, вівся пошук додаткових бібліотек і способів інтеграції з існуючими сервісами як для максимальної зручності написання скриптів, так і для відстеження специфічних показників. Паралельно здійснювалося поетапне тестування створених скриптів і їх впровадження в експлуатацію. В підсумку, розроблений моніторинг став невідривної частиною підтримуваних проектів, що забезпечує високу доступність, відмовостійкість і інформацію про стан систем.

В ході використання створеного моніторингу відділами адміністрування і підтримки серверів, розробки програмного забезпечення та підрозділом аналітиків був відзначений ряд позитивних аспектів впливу на проект, а також абсолютно несподівані плюси продукту:

- підвищення відмовостійкості сервісів, в тому числі шляхом інтеграції з ними;

- система дозволяє правильно визначати і планувати шляхи модернізації проекту і його інфраструктури;

- максимально швидко і точно інформує про можливу проблему;

- використання бази даних моніторингу для різних зовнішніх потреб (наприклад, звіти аналітичного відділу);

- простота інтеграції з різними зовнішніми бібліотеками і сервісами;

- автоматизація рутинних завдань системного адміністратора;

- мінімізація часу простою і, як наслідок, підписання SLA, тобто вихід на новий рівень обслуговування клієнтів;

- незалежність від віртуалізації, програмного забезпечення та інших факторів, можливість моніторингу специфічних показників;

- мінімальний час навчання роботи з системою;

- простота делегування прав доступу і дозволів;

- виявлення «багів» і необроблених винятків в коді завдяки звіту про стан сервісів;

- простота настройки під конкретний стан інфраструктури за допомогою вбудованих функцій Nagios і спеціальних параметрів;

- повна історія доступності сервісів і зміни показників;

- простота впровадження і масштабованості.

В цілому пропонований продукт для моніторингу серверів і служб, а також автоматизації завдань і інвентаризації допоможе значно знизити час простою сервісів, підвищити якість послуг, що надаються клієнту компанією, а також значно поліпшити продуктивність.

ПЕРЕЛІК ДЖЕРЕЛ

1. M. Yuffe et al., “A Fully Integrated Multi-CPU, GPU and Memory Controller 32nm Processor,” Proc. IEEE Int’l Solid-State Circuits Conf., 2011; doi:10.1109/ISSCC.2011.5746311.

2. Intel 64 and IA-32 Architectures Software Developer’s Manual, Intel, 2016.

3. F. McKeen et al., “Innovative Instructions and Software Model for Isolated Execution,” Proc. 2nd Int’l Workshop Hardware and Architectural Support for Security and Privacy, 2013; doi:10.1145/2487726. 2488368.

4. P. Hammarlund et al., “Haswell: The Fourth-Generation Intel Core Processor, IEEE Micro, vol. 34, no. 2, 2014, pp. 6–20.

5. E. Rotem, “Intel Architecture, Code Name Skylake Deep Dive: A New Architecture to Manage Power Performance and Energy Efficiency,” Intel Developer Forum, 2015.

6. E. Rotem et al., “Energy Aware Race to Halt: A Down to EARtH Approach for Platform Energy Management,” IEEE Computer Architecture Letters, vol. 13, no. 1, 2014, pp. 25–28.

7. A. Yasin, “Software Optimizations Become Simple with Top-Down Analysis Methodology on Intel Microarchitecture, Code Name Skylake,” Intel Developer Forum, 2015.

8. A. Yasin, “A Top-Down Method for Performance Analysis and Counters Architecture,” Proc. IEEE Int’l Symp. Performance Analysis of Systems and Software, 2014; doi:10.1109/ISPASS.2014.6844459.

9. D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, “An Energy Efficiency Feature Survey of the Intel Haswell Processor,” in IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW), 2015, DOI: 10.1109/IPDPSW.2015.70.

10. Intel 64 and IA-32 Architectures Software Developer’s Manual Volume 3A, 3B, and 3C: System Programming Guide, Intel, Sep2016, order Number: 325384-060US. [Online]. Available:

http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-system-programming-manual-325384.pdf

11. R. Schöne, D. Molka, and M. Werner, “Wake-up Latencies for Processor Idle States on Current x86 Processors,” Computer Science – Research and Development, 2014, DOI: 10.1007/s00450-014-0270-z.

12. R. Schöne, T. Ilsche, M. Bielert, D. Molka, and D. Hackenberg, “Software Controlled Clock Modulation for Energy Efficiency Optimization on Intel Processors,” in Proceedings of the 4th International Workshop on Energy Efficient Supercomputing (E2SC), 2016, DOI: 10.1109/E2SC.2016.15.

13. B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz, “Beyond DVFS: A First Look at Performance under a Hardware-Enforced Power Bound,” in 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum, 2012, DOI: 10.1109/IPDPSW.2012.116.

14. K. Lange, “Identifying Shades of Green: The SPECpower Benchmarks,” Computer, 2009, DOI: 10.1109/MC.2009.84.

15. J. Bucek, K.-D. Lange, and J. v. Kistowski, “SPEC CPU2017: Next-Generation Compute Benchmark,” in Companion of the 2018 ACM/SPEC International Conference on Performance Engineering, DOI: 10.1145/3185768.3185771.

16. A. Fog, “The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers,” online, Technical University of Denmark, Sep 2018. [Online]. Available: http://agner.org/optimize/microarchitecture.pdf

17. N. Kurd, M. Chowdhury, E. Burton, T. Thomas, C. Mozak, B. Boswell, M. Lal et al., “Haswell: A family of IA 22nm processors,” in IEEE International Solid - State Circuits Conference - (ISSCC), 2014, DOI: 10.1109/ISSCC.2014.6757361.

18. B. Bowhill, B. Stackhouse, N. Nassif, Z. Yang, A. Raghavan, C. Morganti et al., “The xeon® processor e5-2600 v3: A 22nm 18-core product family,”

in IEEE International Solid-State Circuits Conference - (ISSCC), 2015, DOI: 10.1109/ISSCC.2015.7062934.

19. Intel 64 and IA-32 Architectures Optimization Reference Manual, Intel, Apr 2018, order Number: 248966-040. [Online]. Available: https://software.intel.com/sites/default/files/managed/9e/bc/ 64-ia-32-architectures-optimization-manual.pdf

20. S. M. Tam, H. Muljono, M. Huang, S. Iyer, K. Royneogi, N. Satti, R. Qureshi et al., "SkyLake-SP: A 14nm 28-Core Xeon Processor," in IEEE International Solid - State Circuits Conference - (ISSCC), DOI: 110.1109/ISSCC.2018.8310170.

21. "Intel Xeon Processor Scalable Family Technical Overview," Jul 2017. [Online]. Available: https://software.intel.com/en-us/articles/ intel-xeon-processor-scalable-family-technical-overview

22. A. Mazouz, A. Laurent, B. Pradelle, and W. Jalby, "Evaluation of CPU Frequency Transition Latency," Computer Science - Research and Development, 2014, DOI: 10.1007/s00450-013-0240-x.

23. "Advanced configuration and power interface (acpi) specification, revision 6.3," Jan. 2018, online at uefi.org (accessed 2019-03-27).

24. T. Ilsche, R. Schöne, P. Joram, M. Bielert, and A. Gocht, "System Monitoring with lo2s: Power and Runtime Impact of C-State Transitions," in IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2018, DOI: 10.1109/IPDPSW.2018.00114.

25. Energy Efficient Servers: Blueprints for Data Center Optimization. D. Hackenberg, R. Oldenburg, D. Molka, and R. Schöne, "Introducing FIRESTARTER: A processor stress test utility," in International Green Computing Conference (IGCC), 2013, DOI: http://dx.doi.org/10.1109/IGCC.2013.6604507.

26. T. Ilsche, R. Schöne, M. Bielert, A. Gocht, and D. Hackenberg, "lo2s — Multi-core System and Application Performance Analysis for Linux," in IEEE International Conference on Cluster Computing (CLUSTER), 2017, DOI: 10.1109/CLUSTER.2017.116.

27. A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer et al., “Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir,” in Tools for High Performance Computing 2011, 2012, DOI: 10.1007/978-3-642-31476-6_7.

28. Правила безпечної експлуатації електроустановок споживачів [Електронний ресурс] // Міністерство праці та соціально політики України. – 1998. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0093-98#Text.

29. Вимоги безпеки під час експлуатації обчислювальної техніки – Київ: Національний технічний університет України КПІ, 2013. – 30 с.

30. Охорона праці в галузі [Електронний ресурс] – Режим доступу до ресурсу: https://tiphaman.top/book_ohorona-praci-v-galuzi_876/.

31. Правила пожежної безпеки в Україні [Електронний ресурс] // Міністерство внутрішніх справ України. – 2015. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0252-15#Text.

32. Основи охорони праці [Електронний ресурс]. – 2003. – Режим доступу до ресурсу: https://library.if.ua/book/86/6031.html.

33. Варивода К.С. Вплив комп'ютера на психофункціональний стан користувача. // Переяслав-Хмельницький державний педагогічний університет імені Григорія Сковороди. – 10 с.

34. Державні санітарні правила і норми влаштування, утримання загальноосвітніх навчальних закладів та організації навчально-виховного процесу [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0063588-01#Text.

35. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508-18#Text.

36. Бедрій Я. Основи охорони праці користувачів персональних комп’ютерів / Ярослав Бедрій. – Київ: Богдан, 2014. – 144 с.

37. Коцур Н. Безпека життєдіяльності школярів під час роботи з комп'ютерною технікою: Медико-психологічні аспекти / Н. Коцур, Ю. Гріненко. – 2014. – 9 с.

38. “Here’s How Much Energy All US Data Centers Consume,” *DataCenter Knowledge*, 27-Jun-2016. [Online]. Available:https://tinyurl.com/y96cy9rb. [Accessed: 03-Oct-2018].

39. “Intel® 64 and IA-32 Architecture’s Software Developer’s Manual:Vol. 3B,” *Intel*. [Online]. Available: https://tinyurl.com/y9rcq29c.[Accessed: 19-Mar-2018].

40. H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S.McKinley, “Looking Back on the Language and Hardware

ДОДАТКИ

ДОДАТОК А

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

Компонент-менеджер для аналіза зібраних даних моніторингу серверів

```
from mod_python import apache , util ;

def index ( req ) :
  req.content_type = " text / html "
  text_str = [ ]
  req.add_common_vars ( )
  if req.form.has_key (' host_name ' ): # Перевіряємо чи прийшов даний
параметр .

    timestamp = req.form [' timestamp '] # Отримуємо тимчасову мітку
    text_str.append ( "[" + timestamp + "] PROCESS_SERVICE_CHECK_RESULT " )
    req.write ("\ n timestamp =" + timestamp )

    host_name = req.form [' host_name '] # Отримуємо ім'я хоста
    text_str.append ( host_name )
    req.write ("\ n host_name =" + host_name )

    description = req.form [' description '] # Отримуємо назва перевірки
    text_str.append ( description )
    req.write ("\ n description =" + description )

    return_code = req.form [' return_code '] # Отримуємо код завершення
перевірки
    text_str.append ( return_code )
    req.write ("\ n return_code =" + return_code )

    plugin_output = req.form [' plugin_output '] # Отримуємо результати
перевірки
    text_str.append ( plugin_output )
    req.write ("\ n plugin_output =" + plugin_output )

    text = '; ' . join ( text_str ) # Формуємо рядок і записуємо в файл
    file = open ( '/ usr / share / nagios / hdtp / hosts / ' + host_name , '
a +' )
    file.write ("\ n " + text )
    file.close ( )
```

Компонент-агент для збору даних на спостережуваному сервері

```
#! / usr / bin / python
# - * - Coding : utf -8 - * -
# Завантажуємо бібліотеки
import os
import sys
import urllib
import re
import time
import datetime
import subprocess
```

```
class dataTransferPlugin : # Основний клас
    def __ init__ ( self ): # Конструктор класу
        self._nagios_server =
        self._nagios_plugins_dir =
        self._host_name =
        self._method =
        _config = self.getConfig ( )
        self.letsStart ( _config )

    def makeCheckCommand ( self , data = None ): # Функція виконання
        if data : виконує запуск перевірок

            scriptResult = [ ]
            _scriptResult = subprocess.Popen ( self._nagios_plugins_dir +
"/" + data + " ; echo '> ' $ ? " , shell = True , stdout = subprocess.PIPE )
            _scriptResult = _scriptResult.communicate ( ) [ 0 ]
            _scriptResult = _scriptResult.replace ("\ n " , )
# Обробка результатів перевірок
            plugin_output = re.search ( ur " [ \ W | \ D | \ w | \ d ] * ( ?
! => )", _scriptResult )
            if str ( plugin_output ) ! = ' None ' :
                scriptResult.append ( plugin_output.group ( ))
            plugin_output = re.search ( ur "( ? <=> ) [ \ d ] { 1 } " ,
_scriptResult )
            if str ( plugin_output ) ! = ' None ' :
                scriptResult.append ( plugin_output.group ( ))
            return scriptResult

    def killPid ( self , data = None ): # Функція вбиває всі перевірки по
        if data : счетсчіку .
            scriptResult = os.popen ( " kill -9 " + str ( data )) . read ( )
            return scriptResult
# Функція управління методом передачі даних.
    def dataTransfer ( self , command , command_name , method = ' GET ' ) :
        scriptResult = self.makeCheckCommand ( command )
        if method.upper ( ) == ' GET ' :
            self.httpGETRequest ( scriptResult , command_name )
        if method.upper ( ) == ' POST ' :
            self.httpPOSTRequest ( scriptResult , command_name )
# Функція передачі даних на сервер моніторингу GET запитом.
    def httpGETRequest ( self , data = None , data_name = None ) :
        if data ! = None :
            description = ;
            if data ! = None :
                description = data_name
            timestamp = time.strftime ( " % s " , time.localtime ( ))
            print " GET : [" + str ( timestamp ) + "] ; " + self._host_name
+ " ; " + description + " ; " + data [ 1 ] + " ; " + data [ 0 ]
            params = urllib.urlencode ( {' timestamp ' : timestamp , '
host_name ' : self._host_name ,
                                         ' description ' : description , '
return_code ' : data [ 1 ] , ' plugin_output ' : data [ 0 ]} )

            f = urllib.urlopen ( self._nagios_server + " % s " % params )
            f.close ( )
# Функція передачі даних на сервер моніторингу POST запитом.
    def httpPOSTRequest ( self , data = None ) :
        if data ! = None :
            description = ;
            if data ! = None :
```

```
                description = data_name
            timestamp = time.strftime ( " % s " , time.localtime ( ))
            print " POST : [" + str ( timestamp ) + "] ; " + self._host_name
+ " ; " + description + " ; " + data [ 1 ] + " ; " + data [ 0 ]
            params = urllib.urlencode ( {' timestamp ' : timestamp , '
host_name ' : self._host_name ,
                                         ' description ' : description , '
return_code ' : data [ 1 ] , ' plugin_output ' : data [ 0 ]} )

            f = urllib.urlopen ( self._nagios_server , params )
            f.close ( )
# Функція видалення спецсимволов
    def delSpecSymbols ( self , data ) :
        replace_this = re.compile ( r " [ \ n | \ t ] + " )
        clearData = replace_this.sub ( "" , data )
        return clearData.strip ( )
# Функція читання конфігураційних файлів.
    def getConfig ( self ) :
        f = open ( r '/ etc / hdtp / hdtp.conf ' )
        number = 0
        index = {}
        line = f.readlines ( )
        for l in line :
            search_host_name = re.search ( ur "( ? <= host_name ) * [ \ D |
\ d | \ . ] * " , l )
            if str ( search_host_name ) ! = ' None ' :
                host_name = self.delSpecSymbols ( search_host_name.group (
))
                self._host_name = host_name
                continue
            search_nagios_server = re.search ( ur "( ? <= nagios_server ) *
http:// [ \ D | \ d ] * " , l )
            if str ( search_nagios_server ) ! = ' None ' :
                nagios_server = self.delSpecSymbols (
search_nagios_server.group ( ))
                self._nagios_server = nagios_server
                continue
            search_nagios_plugins_dir = re.search ( ur "( ? <=
nagios_plugins_dir ) * [ \ D | \ d ] * " , l )
            if str ( search_nagios_plugins_dir ) ! = ' None ' :
                nagios_plugins_dir = self.delSpecSymbols (
search_nagios_plugins_dir.group ( ))
                self._nagios_plugins_dir = nagios_plugins_dir
                continue
            search_lifetime = re.search ( ur "( ? <= lifetime ) * \ d * " ,
l )
            if str ( search_lifetime ) ! = ' None ' :
                lifetime_data = self.delSpecSymbols ( search_lifetime.group
( ))
                self._max_command_lifetime = lifetime_data
                continue
            search_comment = re.search ( ur "( ? <= # ) \ D * " , l )
            if str ( search_comment ) ! = ' None ' :
                continue
            search_method = re.search ( ur "( ? <= method ) * \ D { 3,5 } "
, l )
            if str ( search_method ) ! = ' None ' :
                method_data = self.delSpecSymbols ( search_method.group ( ))
                self._method = method_data
                continue
            search_command_name = re.search ( ur "( ? <= description ) [ | a
```

```
- zA - Z | 0-9 ] * ( ? = command )", l )
            if str ( search_command_name ) ! = ' None ' :
                command_data_name = self.delSpecSymbols (
search_command_name.group ( ))
                number + = 1
                index [ number ] = {}
                index [ number ] [' name '] = command_data_name
            search_command = re.search ( ur "( ? <= command ) [ | \ D | 0-9
] * " , l )
            if str ( search_command ) ! = ' None ' :
                command_data = self.delSpecSymbols ( search_command.group (
))
                index [ number ] [' command '] = command_data
                continue

        f.close ( )
        return index
# Функція паралельного запуску програм перевірок .
    def letsStart ( self , config = {} ) :
        pid = {}
        for line in config : # Основний цикл під процесів
            pid [ line ] = os.fork ( )
            if pid [ line ] :
                pass
            else :
                result = self.dataTransfer ( config [ line ] [' command '] ,
config [ line ] [' name '] , self._method )
                sys.exit ( 0 )
        time.sleep ( float ( self._max_command_lifetime ))
        for i in pid :
            os.kill ( pid [ i ] , 9 )
        sys.exit ( 0 )

plugin = dataTransferPlugin ( )
```

Інструмент для швидкого додавання нових серверів в систему управління

```
# hdtp - hosts
# Приклад команд на додавання хоста і сервісу через систему ym .
# ym - c add - host - o photo.auditory.ru - p use = linux - server - p
contact_groups = admins
# ym - c add - service - o PING - d www.auditory.ru - p use = generic -
service - p check_command = check_ping
DIR = '/ var / nagios / hdtp / ' # Папка містить файли -хости з даними.
NAGIOS_ETC_DIR = '/ etc / nagios ' # Основний конфігураційний каталог Nagios
HOSTS_CFG = '$ { NAGIOS_ETC_DIR } / objects / hosts.cfg ' # Файл з хостами
CACHE = '/ var / hdtp - search / new - hosts.cache ' # Файл з тимчасовими
даними
mkdir - p / var / hdtp - search /
function save2cache {# Функція формування команд та збереження в тимчасовому
файлі.
addhost = ` echo " ym - c add - host - o $ 1 - p use = linux - server - p
contact_groups = admins " `
echo $ addhost ;
echo $ addhost >> $ CACHE
echo " $ 2 " | while read service ; do
        addservice = ` echo " ym - c add - service - o $ service - d $ 1 - p
use = generic - service - p check_command = check_nodata " `
        echo $ addservice ; echo $ addservice >> $ CACHE
```

```
done
}
function check_new {# Функція аналізу нових даних
        ls $ DIR | while read host ; do
                check_exist = ` grep $ host $ HOSTS_CFG | wc - l `
                if [" $ host " ! = " cmd_file "] && [$ check_exist == 0 ] ;
then
                host_data = ` cat $ DIR $ host | awk - F" ; " ' {
                                srv_list [$ 3 ] = $ 3 ;
                                } END {
                                for ( name in srv_list )
                                                if ( srv_list [ name ] ! =
"" ) print srv_list [ name ] ;
                                } ' `

                save2cache $ host " $ host_data "
                fi
        done
}
check_new # Запуск нового аналізу
# Запит дії від користувача
read - n1 - p " Do you want make all cmd [ Y | N ] ? "
echo ""
case $ REPLY in
  n | N )
  echo " You didn't want to continue ..."
  ; ;
  y | Y )
  echo " Start make commands ..."
  # / bin / bash $ CACHE
  cat $ CACHE
  ; ;
esac
rm $ CACHE
exit 0
```