Plag

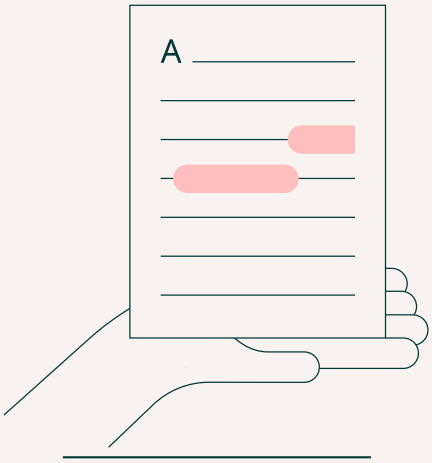
Звіт про оригінальність

● Оцінка схожості 8%

● Ризик плагіату НАЙВИЩИЙ

Olga Kagalo 2025-06-19 22:59

Посилання на звіт: 10mCJ / Посилання користувача: qEAc



Ось вона –
Ваша звіт про оригінальність!

Ми раді повідомити, що перевірка вашого документа завершена, і результати вже готові! Наші алгоритми старанно працювали, щоб знайти збіги в наших базах даних.

На наступних сторінках ви знайдете результати перевірки:

Бали
Збіги
Посилання

Ваш документ було перевірено за такими джерелами:

- ✓ База даних інтернет-джерел
- ✓ База даних наукових статей
- ○ Глибока перевірка (наш вдосконалений алгоритм)

Бали

Збіги тексту	8%
Перефразування	0%
Цитований текст	0%
Неправильне цитування	0%
Збігів не знайдено	92%

Ризик плагіату НАЙВИЩИЙ

Ризик плагіату вказує, як збіги тексту розподілені по документу. Вищий ризик виникає, коли збіги з'являються близько один до одного, наприклад, у тому самому абзаці або розділі.

Оцінка схожості 8%

Оцінка схожості показує, скільки слів або символів у вашому документі збігаються з текстами інших документів, включаючи перефразовані тексти або неправильні цитати.

Збіги

ВСТУП

У сучасному світі ефективне функціонування інформаційних систем є критично важливим для будь-якої організації. Зростаюча складність інфраструктур, що включають численні сервери, мережеві пристрої та додатки, вимагає надійних інструментів для постійного моніторингу та управління продуктивністю. Відсутність своєчасного виявлення проблем може призвести до збоїв, простоїв та значних фінансових втрат. Саме тому розробка комплексної системи моніторингу стає невід'ємною частиною стратегії забезпечення стабільності та доступності IT-сервісів.

Ця робота присвячена розробці інтегрованої системи моніторингу, яка поєднує потужні можливості двох провідних open-source рішень: Ganglia для збору та візуалізації метрик продуктивності кластерів, та Nagios для моніторингу доступності сервісів та оповіщення про критичні події. Комбінований підхід дозволить отримати повне уявлення про стан інфраструктури, починаючи від детальних показників завантаження ресурсів і закінчуючи статусом критично важливих бізнес-сервісів. Ми розглянемо архітектуру системи, етапи її реалізації, а також переваги такого інтегрованого рішення для забезпечення безперебійної роботи складних розподілених середовищ.

1 АНАЛІЗ КЛАСТЕРНИХ СИСТЕМ

Кластерні системи

Кла́стер — це декілька незалежних обчислювальних машин, що використовуються спільно і працюють як одна система для вирішення тих чи інших задач, наприклад, для підвищення продуктивності, забезпечення надійності, спрощення адміністрування тощо. Обчислювальний кластер потрібен для збільшення швидкості обрахунків за допомогою паралельних обчислень, як показано на рисунку 1.1.

Рисунок 1.1 – Кластер

Один з перших архітекторів кластерної технології Грегорі Пфістер дав кластеру наступне визначення: «Кластер — це різновид паралельної або розподіленої системи (рисунок 1.2), яка складається з декількох зв'язаних між собою комп'ютерів і

використовується як єдиний, уніфікований комп'ютерний ресурс».

Рисунок 1.2 - Структура розподіленого кластера

Зазвичай розрізняють наступні основні види кластерів:

Відмовостійкі кластери (High-availability clusters, HA, кластери високої доступності)

Кластери з балансуванням навантаження (Load balancing clusters)

Обчислювальні кластери (High perfomance computing clusters)

Grid-системи

В інформаційних технологіях використовуються також наступні визначення поняття «кластер»:

Кластер — група комп'ютерів, об'єднаних високошвидкісними каналами зв'язку, і що представляє з точки зору користувача єдиний апаратний ресурс.

Кластер — група серверів, об'єднаних логічно, здатних обробляти ідентичні запити, і що використовуються як єдиний ресурс.

Кластер — об'єкт, що забезпечує фізично об'єднане зберігання даних з різних таблиць для прискорення виконання складних запитів, використовуваний в Oracle Database.

Обчислювальний кластер — це масив серверів, об'єднаних деякою комунікаційною мережею. Кожний обчислювальний вузол має свою оперативну пам'ять і працює під керуванням своєї операційної системи.

Найпоширенішим є використання однорідних кластерів, тобто таких, де всі вузли абсолютно однакові по своїй архітектурі й продуктивності.

Для кожного кластера є виділений сервер — керуючий вузол (frontend). На цьому комп'ютері встановлене програмне забезпечення, яке активізує обчислювальні вузли при старті системи й управляє запуском програм на кластері.

Власне обчислювальні процеси користувачів запускаються на обчислювальних вузлах, причому вони розподіляються так, що на кожний процесор доводиться не більш одного обчислювального процеса.

Користувачі мають домашні каталоги на сервері доступу — шлюзі (цей сервер забезпечує зв'язок кластера із зовнішнім світом через корпоративну ЛОМ або Інтернет), безпосередній доступ користувачів на керуючий вузол виключається, а

доступ на обчислювальні вузли кластера можливий (наприклад, для ручного керування компіляцією завдання) (рисунок.1.3).

Рисунок 1.3 - Термінальний кластер

1 Обчислювальний кластер, як правило, працює під керуванням однієї з різновидів ОС Unix багатокористувацької багатозадачної мережевої операційної системи. Зокрема, в ІК НАН України кластери працюють під керуванням ОС Linux вільно розповсюджуваного варіанта Unix.

1 Unix 3 має ряд відмінностей від Windows, яка звичайно працює на персональних комп'ютерах, зокрема ці відмінність стосуються інтерфейсу з користувачем, роботи із процесами й файлової системи.

1 Існує декілька способів зайняти обчислювальні потужності кластера:

1 Запускання багатьох однопроцесорних завдань. Це може бути сприятливим варіантом, якщо потрібно провести багато незалежних обчислювальних експериментів з різними вхідними даними, причому час проведення кожного окремого розрахунків не має значення, а всі дані розміщаються в об'ємі пам'яті, доступному одному процесу.

1 У сучасному інформаційному просторі, де вимоги до обчислювальної потужності, надійності та масштабованості постійно зростають, кластерні системи відіграють ключову роль. Вони є фундаментальною архітектурою для побудови високопродуктивних, відмовостійких та масштабованих інформаційних систем. Цей розділ присвячений детальному аналізу концепції кластерних систем, їхніх основних типів та формулюванню завдання на проектування системи моніторингу для таких середовищ.

Кластерна система – це сукупність незалежних комп'ютерів (вузлів або нод), об'єднаних у єдину логічну систему за допомогою високошвидкісної мережі та спеціального програмного забезпечення. Головна мета створення кластерів полягає у підвищенні загальної продуктивності, доступності та масштабованості, а також забезпеченні відмовостійкості, що неможливо або економічно недоцільно досягти за допомогою одного потужного сервера.

Основні характеристики кластерних систем:

Масштабованість (Scalability): Можливість збільшення обчислювальної потужності або обсягу оброблюваних даних шляхом додавання нових вузлів без суттєвої зміни архітектури системи. Розрізняють вертикальну (збільшення ресурсів одного вузла) та горизонтальну (додавання вузлів) масштабованість, причому кластери орієнтовані

переважно на горизонтальну.

4 Відмовостійкість (Fault Tolerance) / Висока доступність (High Availability): Здатність системи продовжувати 4 функціонувати навіть при 4 відмові одного або декількох її компонентів (вузлів, мережевих з'єднань, дисків). Це досягається за рахунок надмірності (резервування) та механізмів автоматичного переходу на резервні ресурси (failover).

Балансування навантаження (Load Balancing): Розподіл вхідних запитів або обчислювальних завдань між кількома вузлами кластера для рівномірного використання ресурсів та запобігання перевантаженню окремих компонентів.

Економічна ефективність: Використання великої кількості стандартних, відносно недорогих комп'ютерів замість одного надпотужного сервера часто є більш економічно вигідним рішенням.

Розподілена обробка: Завдання можуть бути розбиті на дрібніші частини, які виконуються паралельно на різних вузлах, значно прискорюючи обчислення.

Ключові компоненти кластерної системи:

Вузли (Nodes): Окремі комп'ютери, що входять до складу кластера. Кожен вузол має власні процесори, пам'ять та дисковий простір.

Мережа взаємодії (Interconnect Network): Високошвидкісна мережа (наприклад, Gigabit Ethernet, InfiniBand), що забезпечує швидкий обмін даними 4 між вузлами кластера.

4 Програмне забезпечення кластера (Cluster Software): Включає операційні системи, засоби управління ресурсами, програмне забезпечення для балансування навантаження, забезпечення відмовостійкості, розподілених файлових систем та планувальники завдань.

Загальне сховище даних (Shared Storage): У деяких типах кластерів використовується спільне сховище, доступне для всіх вузлів, що забезпечує узгодженість даних та спрощує перемикання при відмові.

1.2. Характеристика різних типів кластерів

Кластерні системи класифікуються за їхнім основним призначенням та функціональністю:

Кластери високої доступності (High Availability / HA Clusters)

Призначення: Забезпечення безперервної роботи критично важливих сервісів та

додатків 14 шляхом усунення єдиної точки відмови.

14 Принцип роботи: У разі відмови основного вузла (або сервісу на ньому), система автоматично переключає навантаження на резервний (активний/пасивний або активний/активний) вузол, який бере на себе його функції. Цей процес називається "failover".

Типові застосування: Бази даних, веб-сервери, поштові сервери, ERP-системи, файлові сервери.

Ключові технології: Heartbeat, Pacemaker, Corosync, Microsoft Cluster Server (MSCS), Veritas Cluster Server.

Особливості: Зазвичай вимагають спільного сховища даних, щоб усі вузли мали доступ до однієї копії даних. Механізми "fencing" (відключення вузла, що вийшов з ладу, від спільного сховища) для уникнення розбіжностей у даних (split-brain).

13 Кластери балансування навантаження (Load Balancing Clusters)

Призначення: Розподіл вхідних запитів між кількома вузлами кластера для оптимізації використання ресурсів, підвищення продуктивності та забезпечення масштабованості.

Принцип роботи: Вхідні запити надходять до балансувальника навантаження (Load Balancer), який, використовуючи певні алгоритми (наприклад, round-robin, least connections), направляє їх на найменш завантажений або найбільш підходящий вузол.

Типові застосування: Високонавантажені веб-сайти, FTP-сервери, DNS-сервери, проксі-сервери, розподілені мережеві сервіси.

Ключові технології: Nginx, HAProxy, F5 BIG-IP, Cisco ACE, LVS (Linux Virtual Server), облачні балансувальники навантаження (AWS ELB, Google Cloud Load Balancing).

Особливості: Кожен вузол обробляє свою частину навантаження незалежно. Забезпечується легка горизонтальна масштабованість.

Кластери високопродуктивних обчислень 8 (High Performance Computing / HPC Clusters)

Призначення: Виконання складних обчислювальних завдань, що вимагають значних обчислювальних ресурсів, таких як наукові симуляції, моделювання, обробка великих масивів даних.

Принцип роботи: Завдання розбиваються на паралельні підзадачі, які виконуються

одночасно на багатьох вузлах кластера. Для координації використовується спеціальне програмне забезпечення та бібліотеки для паралельних обчислень.

Типові застосування: Метеорологічні прогнози, фінансове моделювання, біоінформатика, інженерні розрахунки (CAD/CAE), криптографія.

Ключові технології: 8 MPI (Message Passing Interface), OpenMP, Slurm, PBS/Torque, Rocks Cluster Distribution.

Особливості: Дуже високі вимоги до швидкості мережі взаємодії (InfiniBand, RoCE), ефективні системи управління чергами завдань та планувальники.

Кластери для розподілених файлових систем та великих даних (Distributed File Systems / Big Data Clusters)

Призначення: Зберігання та обробка надзвичайно великих обсягів даних (петабайти та більше), що не вміщуються на одному сервері, та виконання аналітичних завдань над цими даними.

Принцип роботи: Дані розподіляються та реплікуються між багатьма вузлами кластера. Обчислення виконуються на вузлах, де зберігаються відповідні дані (принцип "переміщення обчислень до даних"). Це мінімізує мережевий трафік.

Типові застосування: Великі дані (Big Data), бізнес-аналітика, машинне навчання, розподілені бази даних, зберігання логів, потокова обробка даних.

Ключові технології: Apache Hadoop (HDFS, YARN), Apache Spark, Apache Cassandra, Apache Kafka, Elasticsearch, Kubernetes (як платформа для розгортання розподілених систем).

Особливості: Висока відмовостійкість за рахунок реплікації даних, лінійна масштабованість, складні системи управління ресурсами та планування завдань.

1.3. Постановка завдання на проектування

З огляду на зростаючу складність та критичну важливість кластерних систем для сучасного бізнесу та науки, ефективний моніторинг стає невід'ємною частиною їхнього життєвого циклу. На відміну від моніторингу окремих серверів, кластерні системи вимагають комплексного підходу, який враховує взаємодію багатьох компонентів, розподіленість даних та обчислень, а також динамічну зміну стану вузлів та сервісів.

Основні виклики та вимоги до моніторингу кластерних систем:

Масштаб: Необхідність моніторингу сотень або тисяч вузлів, що генерують величезний

обсяг метрик.

Розподіленість: Потреба у зборі даних з багатьох джерел та їх агрегації для отримання цілісної картини.

Комплексність: Моніторинг не тільки апаратних ресурсів (CPU, RAM, диск, мережа), але й програмних компонентів, сервісів кластера, процесів, черг завдань та специфічних метрик додатків.

Динамічність: Кластери можуть змінювати свій склад (додавання/видалення вузлів), а навантаження може швидко змінюватися, що вимагає гнучкої та адаптивної системи моніторингу.

Відмовостійкість моніторингу: Сама система моніторингу повинна бути надійною і доступною, щоб забезпечувати контроль навіть у випадку часткових збоїв кластера.

Візуалізація та аналіз: Можливість візуалізувати великі обсяги даних у зрозумілому форматі (графіки, дашборди) та виконувати їхній аналіз для виявлення тенденцій та аномалій.

Оповіщення: Наявність гнучкої системи оповіщень про критичні події, порогові значення або збої для швидкого реагування адміністраторів.

Виходячи з вищезазначеного, завданням на проектування є розробка та обґрунтування архітектури комплексної системи моніторингу для кластерних систем, яка:

Забезпечує збір широкого спектра метрик продуктивності та стану з усіх вузлів та компонентів кластера.

Виконує ефективну агрегацію та зберігання зібраних даних.

Надає потужні засоби візуалізації для аналізу стану кластера.

Має гнучкий механізм сповіщень про виявлені проблеми та критичні події.

Здатна масштабуватися разом із кластером.

Буде реалізована на базі передових та відкритих технологій, зокрема Ganglia та Nagios, використовуючи їхні сильні сторони для створення синергетичного рішення.

Ця система моніторингу має стати ключовим інструментом для адміністраторів кластерів, дозволяючи їм проактивно управляти ресурсами, швидко діагностувати проблеми та забезпечувати високу доступність.

3. МЕРЕЖЕВИЙ МОНІТОРИНГ

Мережевий моніторинг є наріжним каменем забезпечення стабільності, продуктивності та безпеки будь-якої комп'ютерної інфраструктури, особливо у контексті складних кластерних систем. Він надає адміністраторам необхідні дані для прийняття обґрунтованих рішень, проактивного виявлення проблем та ефективного управління ресурсами.

3.1 Моніторинг комп’ютерних мереж

Моніторинг комп'ютерних мереж – це безперервний процес 10 збору, аналізу та візуалізації даних про стан, продуктивність, доступність та використання ресурсів мережевої інфраструктури та її компонентів. Метою моніторингу є забезпечення безперебійної роботи мережі, своєчасне виявлення вузьких місць, прогнозування потенційних проблем, а також оптимізація продуктивності та безпеки.

Основні цілі та завдання моніторингу комп’ютерних мереж:

Забезпечення доступності (Availability): Контроль за працездатністю мережевого обладнання (маршрутизатори, комутатори, брандмауери), серверів, додатків та сервісів. Виявлення збоїв та простоїв.

Оцінка продуктивності (Performance): Моніторинг ключових показників, таких як завантаження каналів зв'язку, швидкість передачі даних, затримки (latency), пропускна здатність (throughput), використання процесора та пам'яті на мережевих пристроях та серверах. Це дозволяє виявляти вузькі місця, які можуть уповільнювати роботу мережі.

Управління використанням ресурсів (Resource Utilization): Відстеження споживання мережевого трафіку, дискового простору, CPU та RAM на різних вузлах. Це допомагає планувати розширення, оптимізувати конфігурації та ефективніше розподіляти ресурси.

Виявлення аномалій та загроз безпеки (Anomaly Detection & Security): Моніторинг нетипової поведінки мережі або пристроїв, що може вказувати на DoS-атаки, несанкціонований доступ, поширення шкідливого програмного забезпечення або неправильну конфігурацію.

Проактивне управління проблемами (Proactive Problem Management): Завдяки моніторингу адміністратори можуть виявляти ознаки майбутніх проблем (наприклад, зростаюче навантаження, падіння вільного дискового простору) до того, як вони призведуть до відмови сервісів.

Звітність та планування ємності (Reporting & Capacity Planning): Зібрані історичні дані дозволяють генерувати звіти про роботу мережі, 12 аналізувати тенденції та приймати

обґрунтовані рішення щодо модернізації та розширення інфраструктури.

Що саме моніториться у мережі:

Пристрої: Маршрутизатори, комутатори, брандмауери, точки доступу Wi-Fi.

Сервери: Фізичні та віртуальні сервери, їх апаратні компоненти (CPU, RAM, диски, температура) та операційні системи.

Додатки та сервіси: Веб-сервіси (HTTP/HTTPS), бази даних, поштові сервери, DNS-сервери, FTP, SSH та будь-які кастомні додатки.

Мережевий трафік: Обсяги переданих даних, джерела та призначення трафіку, типи протоколів, які використовуються.

Мережеві з'єднання: Доступність мережевих інтерфейсів, стан зв'язку, помилки пакетів.

Методи збору даних:

Для моніторингу комп'ютерних мереж використовуються різні протоколи та методи збору даних:

SNMP (Simple Network Management Protocol): Широко використовується для збору інформації про мережеві пристрої (маршрутизатори, комутатори, сервери) та їхній стан.

ICMP (Internet Control Message Protocol): Використовується для перевірки доступності хостів (ping).

WMI (Windows Management Instrumentation): Для моніторингу систем під управлінням Windows.

SSH/Telnet: Для виконання команд та збору даних з Unix/Linux систем.

Агенти: Спеціальні програми, встановлені на моніторингових хостах, які збирають дані та відправляють їх на центральний сервер моніторингу (наприклад, NRPE для Nagios, gmond для Ganglia).

NetFlow/sFlow/IPFIX: Для детального аналізу мережевого трафіку та його потоків.

Лог-файли: Аналіз системних логів та логів додатків для виявлення подій та помилок.

API: Для взаємодії з хмарними сервісами або специфічними додатками.

2.2. Класифікація засобів моніторингу і аналізу

Засоби моніторингу та аналізу комп'ютерних мереж можна класифікувати за кількома

критеріями:

За об'єктом моніторингу:

Моніторинг мережевого обладнання: Засоби для відстеження стану маршрутизаторів, комутаторів, брандмауерів (наприклад, використання SNMP).

Моніторинг серверів: Контроль за апаратними ресурсами (CPU, RAM, диск), операційною системою та встановленими службами на серверах.

Моніторинг додатків та сервісів: Перевірка доступності та продуктивності конкретних програмних додатків (веб-сервери, бази даних, поштові сервіси).

Моніторинг трафіку та потоків даних: Аналіз обсягів, джерел, призначення та типів мережевого трафіку (NetFlow/sFlow аналізатори).

Моніторинг безпеки (SIEM - Security Information and Event Management): Збір та аналіз подій безпеки з різних джерел для виявлення загроз та інцидентів.

За принципом збору даних:

Агентські системи (Agent-based): Вимагають встановлення спеціального програмного агента на кожному моніторинговому вузлі. Агент збирає дані та передає їх на центральний сервер. Переваги: деталізований збір даних, менше мережеве навантаження на центральний сервер. Недоліки: вимагає інсталяції та підтримки агентів. (Приклади: Nagios NRPE, Zabbix Agent, Ganglia gmond).

Без агентські системи (Agentless): Збирають дані віддалено, використовуючи стандартні протоколи (SNMP, WMI, SSH, ICMP, JMX, API). Переваги: легкість розгортання, не потрібно змінювати конфігурацію моніторингових хостів. Недоліки: може створювати більше мережевого навантаження, обмеженість у зборі деяких метрик. (Приклади: Nagios (через ping/SNMP), PRTG Network Monitor).

За типом розгортання:

Локальні (On-premises): Програмне забезпечення встановлюється та працює на власних серверах компанії. Повний контроль над даними та системою.

Хмарні (Cloud-based / SaaS): Моніторинговий сервіс надається як послуга стороннім провайдером. Немає потреби в обслуговуванні власної інфраструктури моніторингу.

Гібридні: Комбінація локальних та хмарних рішень.

За функціональністю:

Системи збору метрик та візуалізації: Орієнтовані на збір числових показників та представлення їх у вигляді графіків, дашбордів для аналізу тенденцій. (Приклади: Ganglia, Grafana, Prometheus).

Системи перевірки доступності та оповіщення: Основний акцент на перевірці працездатності сервісів/хостів та негайному сповіщенні про відхилення від норми. (Приклади: Nagios, Icinga, UptimeRobot).

Системи управління логами (Log Management): Збір, централізація, індексація та аналіз лог-файлів з різних джерел. (Приклади: ELK Stack (Elasticsearch, Logstash, Kibana), Splunk).

Системи трасування (Tracing/APM - Application Performance Monitoring): Детальний аналіз виконання програмних додатків, відстеження запитів через різні сервіси. (Приклади: Jaeger, Zipkin, Dynatrace, New Relic).

За моделлю ліцензування:

Пропрієтарні (Proprietary/Commercial): Платні комерційні рішення з ліцензіями, зазвичай з підтримкою від розробника.

Відкриті (Open-source): Безкоштовні рішення з відкритим вихідним кодом. Можуть мати активну спільноту та/або платну комерційну підтримку. (Приклади: Nagios, Ganglia, Zabbix, Prometheus, Grafana).

Ці класифікації допомагають зрозуміти різноманіття інструментів та вибрати найбільш підходящі для конкретних завдань моніторингу, зокрема для комплексних кластерних середовищ, де часто виникає потреба в комбінації різних підходів та інструментів.

3. ХАРАКТЕРИСТИКА СИСТЕМ МОНІТОРИНГУ

Для забезпечення ефективного контролю за станом та продуктивністю кластерних систем необхідно використовувати потужні та гнучкі інструменти моніторингу. Серед багатьох доступних рішень особливе місце посідають Ganglia та Nagios – дві відкриті системи, які, хоча й мають різну спеціалізацію, можуть чудово доповнювати одна одну у комплексних середовищах. Цей розділ присвячений детальному розгляду характеристик кожної з цих систем.

3.1. Характеристика Ganglia

Ganglia – це 9 масштабована розподілена система моніторингу, призначена 9 переважно для моніторингу високопродуктивних обчислювальних 9 систем, таких як кластери та ґрід-системи. Її основна сила полягає у зборі, агрегації, зберіганні та візуалізації числових метрик продуктивності та стану великої кількості вузлів.

Основне призначення:

Збір метрик продуктивності (CPU, пам'ять, диски, мережевий трафік, завантаження системи, кількість процесів тощо) з великої кількості серверів/вузлів.

Агрегація та централізоване зберігання цих даних.

Візуалізація історичних та поточних даних у вигляді графіків.

Надання можливості для порівняння метрик між різними вузлами кластера.

Ключові особливості та можливості:

Розподілена архітектура: Ganglia спроектована для роботи в розподілених середовищах, що робить її ідеальною для кластерів.

Масштабованість: Здатна ефективно моніторити кластери, що складаються з сотень і навіть тисяч вузлів, з мінімальним навантаженням на мережу та моніторингові вузли.

Ефективний збір даних: Використовує багатоадресну розсилку (multicast) або unicast для збору даних, що є дуже ефективним з точки зору мережевого трафіку.

Використання RRDtool: Дані продуктивності зберігаються за допомогою RRDtool (Round Robin Database tool), що дозволяє ефективно керувати історичними даними, автоматично ущільнюючи їх з часом для економії дискового простору.

Гнучке додавання метрик: Дозволяє легко розширювати перелік моніторингових метрик за допомогою скриптів.

Веб-інтерфейс: Надає інтуїтивно зрозумілий веб-інтерфейс для перегляду графіків продуктивності, порівняння вузлів та отримання загальної картини стану кластера.

Ієрархічний моніторинг: Підтримує моніторинг декількох кластерів або ґрід-систем з централізованого місця.

Архітектура Ganglia:

Ganglia складається з кількох основних компонентів:

gmond (Ganglia Monitoring Daemon): Легкий демон, який запускається на кожному моніторинговому вузлі кластера. Він відповідає за збір метрик продуктивності з локального вузла та передачу їх за допомогою XML через UDP-мультикаст або TCP-юнікаст на інші вузли кластера та до gmetad.

gmetad (Ganglia Meta Daemon): Демон, що зазвичай запускається на центральному сервері моніторингу. Він отримує дані від одного або кількох gmond демонів (або від інших gmetad демонів, дозволяючи будувати ієрархічні структури моніторингу), агрегує їх та зберігає у внутрішній пам'яті. Запити з веб-інтерфейсу (GWEB) надходять саме до gmetad.

RRDtool (Round Robin Database tool): Не є частиною Ganglia, але тісно інтегрований з нею. gmetad використовує RRDtool для зберігання історичних даних метрик у файлах Round Robin Database (.rrd). RRDtool також відповідає за генерацію графіків на основі цих даних.

GWEB (Ganglia Web UI): Веб-інтерфейс, написаний на PHP, який взаємодіє з gmetad для отримання даних та відображення їх у вигляді графіків та таблиць. Надає зручний спосіб візуалізації стану кластера.

Принцип роботи:

Кожен gmond вузла збирає дані про CPU, пам'ять, мережу, диск тощо. Ці дані періодично (за замовчуванням кожні 15 секунд) розсилаються іншим gmond'ам у кластері (або на gmetad). Центральний gmetad отримує ці дані, агрегує їх, зберігає в RRD-файлах і надає їх GWEB для візуалізації. Користувач може переглядати графіки для окремих вузлів, порівнювати їх, а також бачити агреговані показники для всього кластера.

Переваги Ganglia:

Висока масштабованість: Ефективно працює з великою кількістю вузлів.

Низьке навантаження: Демон gmond дуже легкий, а використання мультикасту мінімізує мережевий трафік.

Гарна візуалізація метрик: Чудово підходить для створення графіків продуктивності та аналізу тенденцій.

Гнучкість: Легко розширюється для збору власних метрик.

Історичні дані: Ефективне зберігання довготривалих історичних даних за допомогою RRDtool.

Недоліки Ganglia:

Відсутність вбудованих оповіщень: Ganglia не має власних механізмів оповіщення (email, SMS) про досягнення порогів або збої. Вона лише збирає та візуалізує дані.

Не є системою "управління подіями": Вона не призначена для моніторингу доступності сервісів чи хостів у реальному часі та не обробляє події.

Складність моніторингу окремих сервісів: Хоча можна додавати метрики сервісів, вона не перевіряє їхню працездатність безпосередньо.

3.2. Характеристика Nagios

Nagios (та її форки, такі як Icinga) – це потужна система моніторингу хостів, сервісів та мережевих пристроїв, орієнтована на перевірку доступності та стану компонентів інфраструктури, а також на негайне оповіщення про будь-які відхилення. Nagios вважається одним із "золотих стандартів" у світі відкритого моніторингу.

Основне призначення:

Моніторинг доступності (up/down) хостів, мережевих пристроїв та сервісів.

Виявлення збоїв у роботі додатків, сервісів або мережевої інфраструктури.

Генерація оповіщень (alerts) адміністраторам про виявлені проблеми.

Надання інструментів для аналізу логів та статистики подій.

Ключові особливості та можливості:

Моніторинг хостів та сервісів: Здатність перевіряти стан практично будь-якого мережевого пристрою або сервісу.

Гнучка система оповіщень: Підтримка оповіщень через email, SMS,PagerDuty та інші кастомні методи. Налаштування ескалації оповіщень.

Плагінна архітектура: Nagios Core сам по собі є "рушієм" для виконання перевірок. Вся функціональність моніторингу реалізується за допомогою зовнішніх плагінів (скриптів), що надає величезну гнучкість. Існують тисячі готових плагінів, і можна легко писати власні.

Конфігурація на основі файлів: Усі об'єкти моніторингу (хости, сервіси, контакти, команди) конфігуруються у текстових файлах, що спрощує версіонування та автоматизацію.

Веб-інтерфейс: Класичний веб-інтерфейс для перегляду поточного стану системи, журналів подій, керування оповіщеннями та простоїв (downtime).

Обробники подій (Event Handlers): Можливість автоматично запускати скрипти у відповідь на зміну стану сервісу або хоста (наприклад, перезапустити сервіс, зібрати

додаткову діагностичну інформацію).

Батьківські/дочірні відносини: Можливість визначення залежностей між хостами, що дозволяє уникнути "шторму" оповіщень, коли відмова одного пристрою (наприклад, маршрутизатора) призводить до недоступності багатьох інших.

Архітектура Nagios:

Основні компоненти Nagios включають:

Nagios Core: Головний "рушій" системи, який читає конфігураційні файли, планує та виконує перевірки, обробляє результати, керує оповіщеннями та веде журнали.

Nagios Plugins: Скрипти (виконувані файли), які виконують фактичні перевірки стану хостів та сервісів. Вони повертають код виходу та рядок тексту, які Nagios Core інтерпретує як стан (OK, WARNING, CRITICAL, UNKNOWN) та опис. Приклади: check_ping, check_http, check_disk.

NRPE (Nagios Remote Plugin Executor): Агент, який встановлюється на віддалених серверах (вузлах кластера). Він дозволяє Nagios Core віддалено виконувати локальні плагіни на цих серверах, збираючи інформацію, яка недоступна через стандартні мережеві протоколи (наприклад, завантаження процесора, використання пам'яті, стан локальних процесів).

Nagios Web UI: Веб-інтерфейс, який відображає поточний стан системи, історичні дані про доступність, журнали подій та дозволяє керувати деякими аспектами моніторингу.

NDOUtils (Nagios Data Out Utilities) / NDOmod: Додатковий модуль, який експортує дані з Nagios Core в базу даних (MySQL, PostgreSQL), що дозволяє використовувати їх для складніших звітів, інтеграції з іншими системами або для побудови розширених веб-інтерфейсів.

Принцип роботи:

Nagios Core періодично запускає плагіни (локально або через NRPE на віддалених вузлах), які перевіряють стан об'єктів. Залежно від результатів перевірок, Nagios оновлює стан об'єкта. Якщо стан змінюється на "WARNING" або "CRITICAL" і це виходить за межі визначених порогів та правил, Nagios ініціює процес оповіщення, надсилаючи повідомлення відповідальним контактам. Всі події та зміни стану логуються.

Переваги Nagios:

Надійне оповіщення: Дуже потужна та гнучка система сповіщень.

Гнучкість через плагіни: Практично безмежні можливості для моніторингу будь-яких параметрів.

Висока доступність: Можливість налаштування надмірності для самого сервера Nagios.

Зріла екосистема: Велика спільнота, багато готових плагінів та документації.

Чітке розмежування станів: OK, WARNING, CRITICAL, UNKNOWN – дозволяє точно класифікувати проблеми.

Недоліки Nagios:

Складність конфігурації: Велика кількість конфігураційних файлів може бути складною для новачків.

Відсутність вбудованої візуалізації метрик: Основний веб-інтерфейс Nagios не призначений для відображення графіків продуктивності. Він показує лише поточний стан та історичну доступність. Для графіків потрібна інтеграція з іншими інструментами (наприклад, pnp4nagios, Grafana).

Масштабованість: Хоча Nagios Core є досить продуктивним, на дуже великих інфраструктурах (тисячі хостів) може вимагати розподілених конфігурацій або використання форків/альтернатив (Icinga, Zabbix).

Не призначений для збору великого обсягу "сирих" метрик: Його архітектура краще підходить для перевірок "так/ні" або "значення_вище/нижче_порогу", ніж для збору мільйонів точок даних за секунду.

4.РОЗРОБКА КОМПЛЕКСНОЇ СИСТЕМИ МОНІТОРИНГУ НА БАЗІ GANGLIA ТА NAGIOS

4.1.Обґрунтування інтеграції Ganglia та Nagios

Ефективний моніторинг кластерних систем вимагає інструментів, які можуть не лише збирати великі обсяги метрик продуктивності, а й оперативно реагувати на критичні події. Як було розглянуто в розділі 3, Ganglia чудово справляється з першим завданням, надаючи глибокий огляд ресурсного навантаження вузлів, але не має вбудованих механізмів сповіщення. Nagios, навпаки, є лідером у сфері перевірки доступності та гнучкого сповіщення, але його вбудовані можливості збору та візуалізації історичних метрик є обмеженими порівняно з Ganglia.

Переваги та недоліки Ganglia як самостійної системи моніторингу кластерів

Переваги Ganglia:

Висока масштабованість та ефективність збору метрик: Розроблена спеціально для кластерів, Ganglia може збирати тисячі метрик з сотень вузлів з мінімальним навантаженням на мережу та систему.

Детальна візуалізація продуктивності: Завдяки інтеграції з RRDtool, Ganglia створює високоінформативні графіки використання CPU, пам'яті, мережі, дискового I/O та інших системних ресурсів, дозволяючи відстежувати тенденції та виявляти вузькі місця.

Агрегація даних: Здатність агрегувати метрики з усього кластера або підгруп вузлів, надаючи загальну картину стану.

Гнучкість у додаванні власних метрик: Легко розширюється за допомогою простих скриптів для моніторингу специфічних для додатків показників.

Недоліки Ganglia:

Відсутність вбудованих механізмів оповіщення: Ganglia не надсилає сповіщень (email, SMS) при досягненні певних порогів або відмові сервісів, що є критично важливим для проактивного реагування.

Обмежений моніторинг доступності сервісів: Хоча вона може збирати метрики сервісів, Ganglia не є інструментом для активної перевірки їхньої працездатності (наприклад, чи відповідає веб-сервер на HTTP-запити).

Не є системою управління подіями: Вона не надає функціоналу для обробки подій, встановлення їх пріоритетів, ескалації або управління простоєм.

Переваги та недоліки Nagios як самостійної системи моніторингу мережевих сервісів та хостів

Переваги Nagios:

Потужна та гнучка система оповіщень: Основна перевага Nagios – це його здатність негайно сповіщати адміністраторів про проблеми через різні канали, з налаштуванням ескалації та періодичності.

Широка плагінна архітектура: Величезна кількість готових плагінів та можливість легко створювати власні дозволяють моніторити практично будь-який аспект IT-інфраструктури.

Моніторинг доступності (Up/Down) та стану сервісів: Nagios відмінно підходить для перевірки, чи доступні хости та чи працюють коректно критичні сервіси (HTTP, SSH, DB, DNS тощо).

Обробники подій (Event Handlers): Можливість автоматично виконувати дії у відповідь на виявлені проблеми (наприклад, спробувати перезапустити сервіс).

Логічне представлення залежностей: Дозволяє налаштовувати батьківсько-дочірні зв'язки для уникнення "шторму" оповіщень.

Недоліки Nagios:

Обмежена візуалізація метрик продуктивності: Основний веб-інтерфейс Nagios не призначений для відображення графіків та історичних даних метрик продуктивності у зручному для аналізу вигляді. Він більше фокусується на поточному стані.

Масштабованість для збору метрик: Хоча Nagios може моніторити багато хостів, збір великих обсягів числових метрик з високою частотою може створювати значне навантаження та вимагати додаткових розширень (таких як NDOutils для зберігання даних).

Складність конфігурації: Конфігурація через текстові файли може бути трудомісткою та схильною до помилок при великій кількості об'єктів.

Інтеграція Ganglia та Nagios дозволяє створити комплексну систему, яка компенсує недоліки однієї системи перевагами іншої:

Ganglia надає Nagios деталізовані метрики: Nagios може використовувати дані, зібрані Ganglia, як джерело для перевірок порогів та генерації оповіщень. Це означає, що Nagios може не тільки перевіряти, чи працює сервіс, але й отримувати сповіщення, якщо, наприклад, завантаження процесора перевищує 80% протягом 5 хвилин, а ці дані надані Ganglia.

Nagios забезпечує Ganglia можливостями оповіщення: Ganglia сама по собі не оповіщає. Інтеграція дозволяє Nagios "забирати" дані з Ganglia, інтерпретувати їх та надсилати сповіщення про виявлені аномалії, які відстежує Ganglia.

Єдина точка контролю та повний огляд: Адміністратори отримують повний огляд стану кластера: Nagios повідомляє про критичні події, а Ganglia дозволяє глибоко зануритися у графіки продуктивності для діагностики та пошуку причин проблем.

Оптимізація ресурсів моніторингу: Уникається дублювання збору метрик. Ganglia ефективно збирає всі числові дані, а Nagios фокусується на їх аналізі для виявлення проблем та оповіщень.

Таким чином, комбінація Ganglia та Nagios є потужним рішенням для моніторингу кластерних систем, що забезпечує як проактивне виявлення проблем, так і глибокий аналіз продуктивності.

4.2 Архітектура інтегрованої системи моніторингу

Архітектура комплексної системи моніторингу на базі Ganglia та Nagios розроблена з урахуванням мінімізації навантаження на кластер, забезпечення відмовостійкості та гнучкості. Вона передбачає розподілену структуру, де кожен компонент виконує свою специфічну функцію.

4.2.1. Загальна схема архітектури

[На цьому місці, у фінальному документі, має бути графічна схема архітектури. Вона повинна відображати наступні елементи:]

Кластерні вузли (Cluster Nodes): Це сервери, що є частиною обчислювального кластера. На кожному з них встановлено gmond та NRPE.

Сервер моніторингу (Monitoring Server): Окремий сервер, на якому розташовуються центральні компоненти Ganglia (gmetad, GWEB) та Nagios (Nagios Core, Nagios Plugins, Web UI, можливо NDOutils).

Мережа кластера (Cluster Network): Високошвидкісна мережа, що з'єднує вузли кластера.

Мережа управління (Management Network): Мережа, що з'єднує сервер моніторингу з кластерними вузлами.

Опис компонентів Ganglia та їх взаємодії

gmond (Ganglia Monitoring Daemon) на кластерних вузлах:

Функція: Збирає локальні метрики продуктивності (CPU, RAM, диски, мережа, завантаження ОС, запущені процеси тощо) на кожному вузлі кластера.

Взаємодія: За замовчуванням gmond використовує UDP мультикаст (порт 8649) для анонсування зібраних метрик іншим gmond'ам у тому ж кластері. Це дозволяє кожному вузлу мати локальну копію актуальних метрик всього кластера. Для централізованого збору також може бути налаштований unicast-режим, де gmond відправляє дані безпосередньо до gmetad.

Надійність: Легкий та ефективний, мінімально впливає на продуктивність вузла.

gmetad (Ganglia Meta Daemon) на сервері моніторингу:

Функція: Агрегує дані, отримані від одного або декількох gmond'ів (або інших gmetad у разі ієрархічного моніторингу). Він є центральною точкою збору та запитів для веб-

інтерфейсу Ganglia.

Взаємодія: gmetad слухає порт 8649 (UDP/TCP) для прийому даних. Після агрегації, він зберігає ці дані у RRD-файлах.

Зберігання: Відповідає за створення та оновлення RRD-файлів, які є базою даних для довгострокового зберігання метрик.

RRDtool (Round Robin Database tool) на сервері моніторингу:

Функція: Низькорівневий інструмент для зберігання та обробки часових рядів даних. Він ефективно управляє історичними даними, автоматично "ущільнюючи" їх (наприклад, з кожні 5 хвилин до кожних 30 хвилин для старих даних), що запобігає нескінченному росту розміру баз даних.

Взаємодія: gmetad використовує RRDtool API для запису та читання даних. Веб-інтерфейс Ganglia (GWEB) викликає RRDtool для генерації графіків на основі даних з RRD-файлів.

GWEB (Ganglia Web UI) на сервері моніторингу:

Функція: Веб-інтерфейс, що надає графічний вигляд зібраних даних. Дозволяє переглядати поточні та історичні графіки продуктивності для окремих вузлів та всього кластера.

Взаємодія: GWEB (як правило, PHP-додаток) звертається до gmetad для отримання поточних даних та до RRD-файлів (через RRDtool) для генерації графіків.

Опис компонентів Nagios та їх взаємодії

Nagios Core на сервері моніторингу:

Функція: Ядро системи моніторингу. Відповідає за виконання перевірок, обробку результатів, керування станами хостів та сервісів, виконання обробників подій та генерацію сповіщень.

Взаємодія: Читає конфігураційні файли, планує перевірки за допомогою плагінів.

Nagios Plugins на сервері моніторингу:

Функція: Зовнішні виконувані скрипти/програми, що виконують фактичні перевірки. Вони можуть перевіряти доступність портів, наявність процесів, значення певних параметрів, навантаження та багато іншого.

Взаємодія: Nagios Core запускає ці плагіни, інтерпретує їхній код виходу (0-OK, 1-WARNING, 2-CRITICAL, 3-UNKNOWN) та стандартний вивід.

NRPE (Nagios Remote Plugin Executor) на кластерних вузлах:

Функція: Легкий агент, який запускається на віддалених Linux/Unix вузлах кластера. Він дозволяє Nagios Core безпечно виконувати локальні плагіни на цих вузлах. Це критично важливо для збору даних, які неможливо отримати віддалено (наприклад, точне використання дискового простору, завантаження процесора за даними ОС).

Взаємодія: NRPE слухає на TCP-порті (зазвичай 5666) запити від Nagios Core. При отриманні запиту, NRPE виконує вказаний плагін на локальній машині та повертає результат Nagios Core.

Nagios Web UI на сервері моніторингу:

Функція: Веб-інтерфейс, що надає поточний стан моніторингових об'єктів, лог-файли, інформацію про оповіщення та звіти про доступність. Дозволяє адміністраторам швидко оцінювати загальний стан системи.

Взаємодія: Безпосередньо зчитує статус-файл Nagios Core та лог-файли.

NDOUtils (Nagios Data Out Utilities) на сервері моніторингу (опціонально, але рекомендовано):

Функція: Експортує всі події та статус-інформацію з Nagios Core у зовнішню базу даних (MySQL або PostgreSQL). Це дозволяє використовувати дані для більш складних звітів, інтеграції з іншими системами, а також для побудови розширених веб-інтерфейсів (наприклад, NagiosQL, Centreon, Opsview).

Взаємодія: Модуль NDOmod завантажується в Nagios Core та перехоплює всі події, відправляючи їх до NDO2DB, який записує їх у базу даних.

Інтеграція Ganglia та Nagios може бути реалізована декількома способами, що дозволяють Nagios використовувати багаті дані Ganglia для прийняття рішень щодо оповіщень:

Використання Ganglia-плагінів для Nagios:

Найпоширеніший та найефективніший метод. Спеціальні плагіни Nagios (наприклад, check_ganglia) встановлюються на сервері моніторингу.

Ці плагіни звертаються до gmetad Ganglia (через його API або шляхом парсингу його XML-виводу) або безпосередньо до RRD-файлів, щоб отримати конкретні метрики

продуктивності (наприклад, використання CPU, обсяг вільної пам'яті, мережевий трафік) для певного вузла або кластера.

Плагін порівнює отримане значення з визначеними порогами (WARNING, CRITICAL) і повертає відповідний статус до Nagios Core.

Переваги: Використовує існуючу інфраструктуру Ganglia для збору даних, не дублюючи зусилля. Nagios отримує доступ до деталізованих метрик.

Приклад використання: Nagios перевіряє check_ganglia_cpu_load для хоста node1. Якщо завантаження перевищує 5.0, генерується WARNING; якщо 10.0 – CRITICAL.

2.Прямий доступ Nagios до RRD-файлів (менш поширений):

Деякі плагіни можуть безпосередньо читати дані з RRD-файлів Ganglia, використовуючи RRDtool бібліотеки.

Переваги: Може бути корисним, якщо gmetad недоступний або для дуже специфічних запитів.

Недоліки: Вимагає від Nagios доступу до файлової системи, де зберігаються RRD-файли, що може бути складніше в налаштуванні безпеки та розподілених середовищах.

3.Використання Nagios Event Handlers для запуску діагностики Ganglia (допоміжно):

При виявленні проблеми Nagios може запустити Event Handler, який, наприклад, згенерує звіт про стан кластера за допомогою Ganglia, або збереже графіки для подальшого аналізу. Це не інтеграція збору даних, а скоріше інтеграція процесів управління.

Таким чином, інтегрована архітектура дозволяє Ganglia бути ефективним "збирачем" та "візуалізатором" числових даних продуктивності, а Nagios – "інтерпретатором" цих даних, перетворюючи їх на оповіщення та керуючи подіями.

4.3. Етапи реалізації комплексної системи

Реалізація комплексної системи моніторингу вимагає послідовного виконання низки кроків, починаючи з планування та закінчуючи тестуванням.

Планування та підготовка

Визначення цілей моніторингу: Чітке розуміння, які параметри необхідно моніторити (CPU, RAM, диск, мережа, процеси, додатки, специфічні для кластера метрики).

Вибір операційної системи: Для сервера моніторингу та кластерних вузлів рекомендовані стабільні дистрибутиви Linux (наприклад, CentOS Stream, Ubuntu Server LTS, Debian). Ці ОС мають добру підтримку для Ganglia та Nagios.

Вимоги до апаратного забезпечення:

Сервер моніторингу: Повинен мати достатньо CPU та RAM для Nagios Core, gmetad, веб-сервера та RRDtool. Залежно від кількості вузлів кластера (N) та частоти збору метрик (M):

CPU: 2-4 ядра (або більше для дуже великих кластерів).

RAM: 4-8 GB (або більше), особливо якщо використовуються NDOutils з базою даних.

Дисковий простір: Значний обсяг для RRD-файлів Ganglia та логів Nagios (кілька десятків/сотень GB, залежно від тривалості зберігання історичних даних). Рекомендовано SSD для RRD-файлів для кращої продуктивності.

Кластерні вузли: Вимоги до апаратних ресурсів для gmond та NRPE мінімальні, оскільки вони дуже легкі.

Підготовка мережевого оточення:

IP-адресація: Забезпечити статичні IP-адреси для сервера моніторингу та всіх кластерних вузлів.

DNS: Налаштувати коректний DNS-резолвінг імен між сервером моніторингу та кластерними вузлами (або використовувати /etc/hosts).

Мережеві порти: Відкрити необхідні порти на брандмауерах (firewalls):

Ganglia: UDP 8649 (для gmond мультикаст), TCP 8649 (для gmond unicast/gmetad), TCP 80/443 (для GWEB).

Nagios: TCP 5666 (для NRPE), TCP 80/443 (для Nagios Web UI).

SSH (22), ICMP (для ping) – для віддаленого управління та базових перевірок.

Синхронізація часу: Вкрай важливо синхронізувати час на всіх вузлах за допомогою NTP (Network Time Protocol) для коректного збору та візуалізації метрик.

Встановлення та конфігурація Ganglia

Встановлення gmond на всіх вузлах кластера:

Виконати інсталяцію пакетів Ganglia Monitoring Daemon через пакетний менеджер ОС (наприклад, sudo apt install ganglia-monitor для Debian/Ubuntu або sudo yum install ganglia-gmond для CentOS).

Конфігурація gmond.conf (файл зазвичай у /etc/ganglia/gmond.conf):

Налаштування секції cluster: name, owner, latlong (опціонально).

Налаштування секції udp_send_channel: Вказати mcast_join (для мультикасту) або host (для unicast до gmetad). Приклад для unicast:

udp_send_channel {

host = monitoring_server_ip

port = 8649

ttl = 1

}

Налаштування секції udp_recv_channel: Якщо gmond також має приймати дані.

Налаштування джерел метрик: Залишити стандартні, або додати кастомні (наприклад, через modules або exec директиви).

Перезапустити службу gmond після внесення змін.

Встановлення та конфігурація gmetad, RRDtool, Apache/Nginx та Web UI на сервері моніторингу:

Встановити пакети ganglia-webfrontend, ganglia-gmetad, rrdtool та веб-сервер (Apache2 або Nginx з PHP-FPM).

Конфігурація gmetad.conf (файл зазвичай у /etc/ganglia/gmetad.conf):

Визначити кластери, які gmetad буде моніторити, вказавши data_source та clustername. Наприклад:

data_source "MyCluster" monitoring_server_ip:8649

(monitoring_server_ip тут означає, що gmetad очікує дані на локальному інтерфейсі, або якщо gmond на вузлах налаштовані надсилати на цей IP). Якщо використовується мультикаст:

data_source "MyCluster" 239.2.11.71:8649

(де 239.2.11.71 – адреса мультикаст-групи).

Налаштувати інтервали оновлення.

Перезапустити службу gmetad.

Налаштування веб-сервера (Apache/Nginx):

Налаштувати віртуальний хост або alias для Ganglia Web UI (зазвичай файли GWEB знаходяться у /usr/share/ganglia-webfrontend).

Переконатися, що веб-сервер може запускати PHP-скрипти.

Перезапустити веб-сервер.

Верифікація роботи Ganglia:

Відкрити веб-браузер та перейти за адресою Ganglia Web UI (наприклад, http://your_monitoring_server/ganglia/).

Переконатися, що з'явилися графіки для всіх вузлів кластера та що дані оновлюються.

Встановлення та конфігурація Nagios

Встановлення Nagios Core, Nagios Plugins на сервері моніторингу:

Встановити пакети nagios4 (або nagios-core), nagios-plugins-basic, nagios-plugins-standard (для Debian/Ubuntu), або аналогічні для CentOS.

Встановити apache2 та libapache2-mod-php (якщо ще не встановлено).

Створити користувача nagiosadmin та призначити пароль: sudo htpasswd -c /etc/nagios4/htpasswd.users nagiosadmin.

Конфігурація основних файлів Nagios (зазвичай у /etc/nagios4/ або /usr/local/nagios/etc/):

nagios.cfg: Основний конфігураційний файл. Переконатися, що він включає всі необхідні конфігураційні файли (наприклад, cfg_dir=/etc/nagios4/objects).

resource.cfg: Визначення змінних, таких як шляхи до плагінів, логін/пароль для віддалених сервісів.

contacts.cfg: Визначити контакти та групи контактів, на які будуть надсилатися оповіщення.

define contact{

contact_name nagiosadmin

use generic-contact

alias Nagios Admin

email admin@example.com

host_notifications_enabled 1

service_notifications_enabled 1

...

}

commands.cfg: Визначення команд, які Nagios буде використовувати для виконання перевірок. Це включає команди для використання NRPE.

6 define command{

6 command_name check_nrpe

6 command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$

6 Встановлення та конфігурація NRPE на кластерних вузлах. Встановити пакет nagios-nrpe-server (або nrpe): sudo apt install nagios-nrpe-server.

Конфігурація nrpe.cfg (зазвичай у /etc/nagios/nrpe.cfg):

Дозволити доступ з IP-адреси сервера моніторингу: allowed_hosts=127.0.0.1, monitoring_server_ip.

Визначити команди, які Nagios може виконувати через NRPE. Це критично важливо для інтеграції з Ganglia. Наприклад:

command[check_users]=/usr/lib/nagios/plugins/check_users -w 5 -c 10

command[check_load]=/usr/lib/nagios/plugins/check_load -w 15,10,5 -c 30,25,20

(Тут ви можете додати також команди, які отримують дані з Ganglia локально на вузлі, якщо це необхідно для специфічних перевірок).

Перезапустити службу nagios-nrpe-server.

Перевірка Nagios:

Перевірити конфігурацію: sudo nagios -v /etc/nagios4/nagios.cfg.

Перезапустити службу Nagios: sudo systemctl restart nagios4.

Відкрити веб-інтерфейс Nagios (http://your_monitoring_server/nagios4 або аналогічно) та увійти під nagiosadmin. Переконатися, що хости та сервіси відображаються коректно.

4.4.Налаштування інтеграції Ganglia та Nagios

Це є найбільш важливою частиною розробки комплексної системи. Вона дозволяє Nagios використовувати дані, зібрані Ganglia, для генерації сповіщень.

Розробка/використання спеціалізованих плагінів Nagios для отримання даних з Ganglia. Існують готові плагіни, наприклад, check_ganglia_metric або check_ganglia_cpu_load. Якщо їх немає, потрібно написати власний скрипт (на Python, Perl, Bash), який:

Звертається до gmetad Ganglia (наприклад, через XML-вивід gmetad або спеціалізований Ganglia-API).

Отримує значення потрібної метрики (наприклад, load_one, mem_free, bytes_in) для конкретного хоста або кластера.

Порівнює отримане значення з визначеними порогами WARNING та CRITICAL.

Повертає відповідний код виходу (0, 1, 2, 3) та інформаційний рядок, зрозумілий Nagios.

Розмістити ці плагіни у директорії плагінів Nagios (наприклад, /usr/lib/nagios/plugins/).

Налаштування визначень хостів та сервісів у Nagios для моніторингу метрик, зібраних Ganglia:

Файл hosts.cfg: Визначити всі вузли кластера як хости 11 Nagios.

11 define host{

11 use linux-server

11 host_name node1.cluster.local

alias Cluster Node 1

address 192.168.1.101

parents router_main

contact_groups admins

check_command check-host-alive

max_check_attempts 5

notification_interval 30

notification_period 24x7

notifications_enabled 1

...

}

Файл services.cfg: Визначити сервіси, які будуть моніторити метрики Ganglia.

Спочатку, додати команду для Ganglia-плагіна до commands.cfg:

define command{

command_name check_ganglia_cpu_load

command_line $USER1$/check_ganglia_metric -h $HOSTADDRESS$ -m load_one -w $ARG1$ -c $ARG2$ -s gmetad_server_ip

}

(Тут check_ganglia_metric - це приклад плагіна, який ви могли написати або знайти. $USER1$ - це шлях до директорії плагінів Nagios, $HOSTADDRESS$ - IP-адреса хоста, $ARG1$ та $ARG2$ - пороги WARNING та CRITICAL, gmetad_server_ip - IP-адреса сервера, де запущено gmetad).

Потім, визначити сервіс для кожного вузла кластера:

define service{

use generic-service

host_name node1.cluster.local

service_description CPU Load (Ganglia)

check_command check_ganglia_cpu_load!5.0!10.0

max_check_attempts 3

normal_check_interval 5

retry_check_interval 1

check_period 24x7

notification_interval 60

notification_period 24x7

notifications_enabled 1

contact_groups admins

...

}

Аналогічно налаштувати сервіси для моніторингу інших метрик Ganglia: пам'ять, диски, мережевий трафік тощо.

Створення кастомних команд у Nagios для використання Ganglia-метричних порогів. Цей пункт вже був частково висвітлений у попередньому, але він підкреслює, що для кожної метрики Ganglia, яку потрібно моніторити за порогами, необхідно створити відповідну команду в Nagios.

Після всіх налаштувань, необхідно знову перевірити конфігурацію Nagios (sudo nagios -v /etc/nagios4/nagios.cfg) та перезапустити службу Nagios (sudo systemctl restart nagios4).

4.5. Моніторинг типових показників кластерної системи

Інтегрована система Ganglia+Nagios дозволяє ефективно моніторити широкий спектр показників кластерної системи.

Моніторинг ресурсів вузлів:

Ganglia: Збирає метрики load_one, load_five, load_fifteen (середнє навантаження за 1, 5, 15 хвилин), cpu_idle, cpu_system, cpu_user тощо. Візуалізує їх у вигляді графіків.

Nagios: Використовує check_ganglia_cpu_load (або аналогічний плагін), щоб отримувати значення load_one (або cpu_idle) з Ganglia та генерувати WARNING/CRITICAL оповіщення, якщо завантаження перевищує встановлені пороги (наприклад, 80% завантаження користувача CPU).

Ganglia: Метрики mem_total, mem_free, mem_cached, mem_buffers, swap_total, swap_free.

Nagios: check_ganglia_memory для сповіщення, якщо вільна пам'ять опускається нижче певного порогу або використання swap-простору надмірне.

Дисковий простір. Ganglia: Метрики disk_total, disk_free, bytes_in, bytes_out для кожного монтованого розділу. Nagios: check_ganglia_disk_usage для оповіщення при низькому вільному місці на критичних файлових системах (наприклад, /, /var, /opt, а також для розподілених файлових систем, таких як HDFS).

Мережевий трафік:

Ganglia: Метрики bytes_in, bytes_out, pkts_in, pkts_out для кожного мережевого інтерфейсу. Візуалізація дозволяє виявляти пікові навантаження та аномалії.

Nagios: Оповіщення про незвичайні рівні трафіку, або якщо мережеві інтерфейси опускаються або генерують забагато помилок (хоча для останнього часто використовуються прямі перевірки SNMP).

Моніторинг процесів та сервісів кластера:

Nagios: Це основний інструмент для моніторингу працездатності критичних сервісів.

Кластерні компоненти: Для кластера Hadoop це можуть бути NameNode, DataNode, ResourceManager, NodeManager, ZooKeeper, Kafka Broker тощо.

Бази даних: MySQL, PostgreSQL, MongoDB (доступність порту, можливість підключення, стан реплікації).

Веб-сервери: Apache, Nginx (HTTP-відповідь, код стану, час відповіді).

Кастомні додатки: Моніторинг за наявністю процесу, за відповіддю API.

NRPE використовується для виконання локальних перевірок на вузлах, наприклад, check_procs для перевірки кількості запущених процесів конкретного сервісу.

Моніторинг стану мережі Nagios:

Доступність вузлів: Базовий check_ping для перевірки мережевої доступності кожного вузла.

Затримки (latency) та втрати пакетів: Моніторинг check_ping може надавати ці метрики, за якими Nagios може оповіщати.

Стан мережевих інтерфейсів: Через SNMP або NRPE Nagios може перевіряти, чи є

інтерфейс "up" та чи немає помилок.

Ganglia: Хоча Ganglia збирає метрики мережевого трафіку, вона не надає функції "alerting" для проблем з мережею, що робить Nagios основним інструментом для сповіщень про мережеві збої.

Налаштування оповіщень у Nagios: Nagios надає потужні та гнучкі можливості для налаштування сповіщень.

Контакти та контактні групи:

Визначення окремих контактів з їхніми email-адресами та/або номерами телефонів.

Об'єднання контактів у групи (наприклад, admins, devops, noc_team).

Приклад у contacts.cfg (як було зазначено вище).

Методи оповіщення:

Email: Стандартний метод, що використовує системний mail команду або зовнішні MUA/MTA (наприклад, Postfix, Sendmail). Налаштовується у commands.cfg та templates.cfg.

SMS: Зазвичай реалізується через Email-to-SMS шлюзи (надаються мобільними операторами) або інтеграцію з SMS-провайдерами через скрипти.

Telegram/Slack/Microsoft Teams: Використання спеціальних плагінів або скриптів, які взаємодіють з API цих платформ. Це дуже популярні методи завдяки їхній інтерактивності та можливості групових чатів.

PagerDuty/Opsgenie: Інтеграція з професійними сервісами управління інцидентами для ротації чергувань та ескалації.

Правила оповіщення:

Період оповіщення: Вказується, коли можуть надсилатися оповіщення (наприклад, 24x7 або workhours).

Інтервал оповіщення: Як часто повторювати оповіщення, якщо проблема не вирішена.

Кількість спроб: Скільки разів Nagios має перевірити стан, перш ніж відправити оповіщення.

Ескалація: Можливість налаштування ескалації оповіщень до вищого рівня підтримки, якщо проблема не вирішується протягом певного часу.

Збереження стану (state_retention_file): Nagios зберігає поточний стан всіх об'єктів для коректної роботи після перезапуску.

Приклад налаштування команди оповіщення в commands.cfg:

2 define command{

2 command_name notify-host-by-email

2 command_line /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" |
2 /usr/bin/mail -s "** 2 $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ **" $CONTACTEMAIL$

}

Ця команда потім призначається контактам або хостам/сервісам.

Використання веб-інтерфейсів:

Інтерфейс Ganglia (GWEB):

Призначення: Основний інструмент для візуалізації метрик продуктивності.

Функціональність:

Огляд кластера: Надає агреговані графіки для всього кластера, показуючи середнє використання CPU, пам'яті, мережевого трафіку.

Деталізація по вузлах: Можливість "заглибитися" у конкретний вузол, щоб переглянути його індивідуальні метрики продуктивності.

Порівняння вузлів: Зручна функція для порівняння метрик декількох вибраних вузлів, що допомагає виявляти відхилення та несправності.

Історичні дані: Перегляд графіків за різні періоди часу (остання година, день, тиждень, місяць, рік).

Кастомні графіки: Можливість створення власних дашбордів або додавання унікальних метрик.

Адміністратори використовують GWEB для аналізу тенденцій, пошуку "гарячих" точок, виявлення аномалій, що не завжди призводять до критичних збоїв, але впливають на

продуктивність.

Інтерфейс Nagios Web UI:

Основний інструмент для моніторингу поточного стану доступності хостів та сервісів, управління оповіщеннями та простоєм.

Функціональність:

Огляд стану: Швидкий огляд стану всіх хостів та сервісів (OK, WARNING, CRITICAL, UNKNOWN).

Деталізація по хостах/сервісах: Перегляд детальної інформації про кожен хост або сервіс, включаючи вихід плагіна, час останньої перевірки, тривалість поточного стану.

Журнал подій: Хронологічний запис усіх подій моніторингу та оповіщень.

Управління простоєм (Downtime): Можливість планувати простої для технічного обслуговування, щоб запобігти генерації зайвих оповіщень.

Звіти: Базові звіти про доступність та продуктивність (performance).

Команди: Можливість вручну перезапустити перевірки, відправити сповіщення, або відключити сповіщення.

Використання: Адміністратори використовують Nagios Web UI для швидкого реагування на проблеми,

Посилання

Це джерела виділених збігів у вашому документі. Кожен збіг позначено темно-зеленим числом, яке відповідає вказаному тут джерелу. Джерела впорядковані за схожістю — чим вищий бал, тим сильніше збіг.

#	Джерело	%
1	uk.wikipedia.org / Комп'ютерний_кластер	5.5%
2	support.nagios.com	0.4%
3	ela.kpi.ua	0.3%
4	duikt.edu.ua	0.3%
5	ekmair.ukma.edu.ua	0.2%
6	thoughtbirds.com	0.2%
7	diit.ust.edu.ua	0.2%
8	doi.org	0.1%
9	elartu.tntu.edu.ua	0.1%
10	science.btsau.edu.ua	0.1%
11	docplayer.es	0.1%
12	eu-conf.com	0.1%
13	jrnl.nau.edu.ua	0.1%
14	guru99.com	0.1%

Plag

Дякуємо, що перевірили
свій документ за допомогою
Plag!