Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Розробка системи збереження даних для хмарної платформи OpenStack

Виконав: студент 6 курсу, групи Сам-61
спеціальності 124 Системний аналіз
(шифр і назва спеціальності)

	(підпис)	Воронін В.С. (прізвище та ініціали)
Керівник	(підпис)	доц. Баран І.О. (прізвище та ініціали)
Нормоконтроль	(підпис)	доц. Мацюк О.В. (прізвище та ініціали)
Завідувач кафедри	(підпис)	доц. Боднарчук І.О. (прізвище та ініціали)
Рецензент	(підпис)	доц. Гладьо Ю.Б. (прізвище та ініціали)

м. Тернопіль – 2020

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
_____ доц. Боднарчук І.О.
(підпис) (прізвище та ініціали)
« » 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 124 Системний аналіз
(шифр і назва спеціальності)

студенту Вороніну Василю Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи збереження даних для хмарної платформи OpenStack

Керівник роботи Баран Ігор Олегович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від «__06__» _листопада_ 2020 року №__4/7-830_

2. Термін подання студентом роботи 21.12.2020

3. Вихідні дані до роботи
наукові літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Аналіз предметної області. 2 Теоретична частина. 3. Практичне дослідження та реалізація системи. 4 Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема роботи. 2. Актуальність. 3. Мета, задачі, об'єкт, предмет дослідження. 4. Особливості платформи OpenStack. 5. Загальна архітектура платформи OpenStack. 6. Види робіт для створення системи. 7. Порівняння основних параметрів розглянутих аналогів. 8. Вимоги до системи, яка розробляється. 9. ПЗ, яке використовувалося. 10. Вільне сховище об'єктів Ceph. 11. Особливості Ceph. 12. Встановлення платформи OpenStack. 13. Скрипт. 14. Розгортання кластера Ceph. 15. Код формування файлу конфігурації та висновків встановлення. 16. Інтеграція хмарної платформи OpenStack з кластером Ceph. 17. Програмні коди функцій для налаштування. 18. Висновки. Основні результати дослідження

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Дмитроца Л. П., доцент	01.11.20	07.12.20
Безпека в НС	Стадник І. Я., професор	01.11.20	09.12.20

7. Дата видачі завдання _____ 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи	06.11.20	Виконано
2	Аналіз літературних джерел	07.11.20-18.11.20	Виконано
3	Обґрунтування актуальності дослідження	18.11-21.11.20	Виконано
4	Аналіз предмету дослідження та предметної області	21.11-26.11.20	Виконано
5	Проведення дослідження методів та засобів аналітичного опрацювання даних	22.11-30.11.20	Виконано
6	Оформлення розділу «Аналіз предметної області»	20.11-26.11.20	Виконано
7	Оформлення розділу «Теоретична частина»	27.11-02.12.20	Виконано
8	Оформлення розділу «Практичне дослідження та реалізація системи»	03.12-10.12.20	Виконано
9	Оформлення розділу «Охорона праці та безпека в надзвичайних ситуаціях»	26.11-12.12.20	Виконано
10	Нормоконтроль	08.12-11.12.20	
11	Попередній захист роботи	11.12.20	Виконано
12	Захист кваліфікаційної роботи	21.12.20	

Студент _____ Воронін В.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Баран І. О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка системи збереження даних для хмарної платформи OpenStack // Кваліфікаційна робота освітнього рівня «Магістр» // Воронін Василь Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, кафедра комп'ютерних наук, група САм–61 // Тернопіль, 2020 // С. – *73* , рис. – *17* , табл.– *4*, слайдів – *18*, додат. – *2*, бібліогр. – *40*.

Ключові слова: CEPH, OPENSTACK, КЛАСТЕР, МАСШТАБОВАНІСТЬ СИСТЕМИ, РЕПЛІКАЦІЯ, СНЕПШОТ

Кваліфікаційна робота присвячена проектуванню та створенню системи збереження даних для хмарної платформи OpenStack.

Для цього проведено аналіз предметної області дослідження, зокрема сформульовано вимоги до розроблюваної системи. Зроблено докладний опис хмарної платформи OpenStack. Проведено порівняльний аналіз окремих систем збереження даних (Ceph, GlusterFS, NDFS, HDFS, pNFS). За результатами порівняння для виконання поставленого завдання на проектування обрано середовище Ceph. Описано архітектуру Ceph, проаналізовано масштабованість і високу доступність моніторів та автентифікації. Докладно описано основні складові динамічного управління кластером та процес чергування даних. Відображено особливості інтеграції хмарної платформи OpenStack з кластером Ceph, зокрема налаштування OpenStack для клієнтів та використання Ceph.

Оскільки Ceph може бути розгорнуте на недорогому апаратному забезпеченні, програмна розробка є якісним дешевшим аналогом сучасним системам збереження даних та має не меншу продуктивність.

ANNOTATION

Data storage system development for cloud platform OpenStack // Master thesis // Voronin Vasyl // Ternopil Ivan Pul'uj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science // Ternopil, 2020 // P. - *73*, Fig. - *17*, Table – *4*, Slide - *18*, References - *40*.

Keywords: CLUSTER, CEPH, OPENSTACK, REPLICATION, SNAPSHOT, SYSTEM SCALABILITY

Thesis deals with the design and creation of a data storage system for the OpenStack cloud platform.

For this purpose the analysis of the subject area of research is carried out, in particular requirements to the developed system are formulated. A detailed description of the OpenStack cloud platform has been made. A comparative analysis of individual data storage systems (Ceph, GlusterFS, NDFS, HDFS, pNFS) is carried out. According to the results of the comparison, the Ceph environment was chosen for the design task. The Ceph architecture is described, the scalability and high availability of monitors and authentication are analyzed. The main components of dynamic cluster management and the process of data alternation are described in detail. Features of integration of the OpenStack cloud platform with the Ceph cluster are shown, in particular settings of OpenStack for clients and use of Ceph.

Software development is a high-quality, cheaper analogue of modern storage systems and has no less performance, because Ceph can be deployed on low-cost hardware.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

Bigdata – структуровані і неструктуровані дані великих обсягів.

Cloud-init – програмне забезпечення для автоматичного налаштування віртуальних серверів при першому запуску.

FUSE (filesystem in userspace) - модуль для ядер UNIX-подібних операційних систем з відкритим вихідним кодом, відноситься до вільного програмного забезпечення, дозволяє користувачам без привілеїв створювати їх власні файлові системи без необхідності переписувати код ядра.

Keyring – ключ доступу

HPC (High Performance Computing) – високопродуктивні обчислення.

POSIX (Portable Operating System Interface for uniX) - набір стандартів, що описують інтерфейси між операційною системою і прикладною програмою (системний API), бібліотеку мови C, набір додатків і їх інтерфейсів.

Split brain – ситуація, як правило, при проблемах мережевого рівня, за якої виникає два набори вузлів будь-якого кластера, що не мають доступу один до одного.

UUID (User Unique IDentifier) – унікальний ідентифікатор користувача.

Баг (Bug) – логічна помилка в програмі.

БД – база даних.

Бекенд (Backend) – програмний код, який відповідає за роботу з сервером або даними.

Демон (Daemon) – процес в Unix системах.

Домен падіння (Failure domain) - набір серверів, в межах яких відбувається відновлення сервісу, при відмові активного вузла.

Ефемерні диски (Ephemeral volumes) – спеціальний тип дискових ресурсів, які видаляються при видаленні віртуального сервера в OpenStack.

Кластер (Cluster) – набір серверів, що працюють як єдине ціле і виконують хостинг кінцевих сервісів.

Образ (Image) – шаблон, спеціально підготовлений диск, що використовується для запуску віртуальних серверів.

ПЗ – програмне забезпечення.

Реплікація (Replication) – можливість одночасного зберігання копій (реплік) на декількох хостах.

РСЗД – розподілена система зберігання даних.

СЗД – система зберігання даних.

Снепшот (Snapshot) – знімок, миттєва копія, заморожений стан файлової системи або блочного пристрою на певний момент часу.

СУБД – система управління базами даних.

ФС – файлова система

Хост (Host) – функціональна одиниця кластера (один сервер або сервіс).

ЗМІСТ

ВСТУП

Актуальність теми. Упродовж останнього часу обчислення за допомогою «хмар» стають основним напрямом для розвитку теперішніх інформаційних технологій (ІТ). Використання хмарних засобів дозволяє значно покращити гнучкість і масштабованість ІТ-інфраструктури підприємства. У минулому часі для розгортання нового бізнес-додатку були потрібні придбання нових серверів, СЗД, купівля прикладного ПЗ та з подальшим його конфігуруванням, при цьому витрати часу на проект складали щонайменше дні, а подеколи і місяці. Зараз ж, з використанням хмарних технологій, розгортання нового програмного засобу відбувається буквально за кілька хвилин. Всі потужності для обчислень та зберігання даних, необхідні для цього, знаходяться у хмарі. Віртуалізація дозволяє об'єднати ресурси багатьох систем та серверів зберігання в єдиний вузол [1].

Відповідно, пришвидшується виділення додаткових ресурсів і дискової ємності вже розгорнутого додатку в періоди пікових навантажень. Непотрібно для цього проводити обслуговування та оновлення серверів і СЗД (встановлювати додаткові вінчестери та процесори, а також модулі оперативної пам'яті), немає необхідності у покупці додаткових систем. Таким чином, програма фактично в «реалі» бере всі необхідні потужності для проведення обчислень з хмари. Отже, при переході на хмарну модель, фірми можуть своєчасно використовувати нові сервіси та динамічно реагувати на появу нових вимог для свого бізнесу. Широкий попит і поширення хмарних технологій, а так само велика увага до систем аналізу великих даних, вимагають використання СЗД, які в змозі впоратися з масовим попитом, з обсягами, що перевищують петабайт, с можливістю масштабування ємності без загроз досягнення будь-яких фізичних обмежень [2].

Таким чином надзвичайно актуальним при повсюдному впровадженні хмарних технологій є принципово новий підхід до побудови СЗД в центрах обробки даних.

Мета роботи полягає у побудові надійної СЗД, яка легко масштабується та інтегрується з OpenStack.

Для досягнення вказаної мети, в роботі поставлено та розв'язано **наступні задачі:**

– проаналізувати доступні Linux/Unix OpenSource РСЗД (включаючи розподілені ФС);

– за результатами проведеного аналізу вибрати ту СЗД, яка найбільш повно задовольняє всім вимогам;

– спроектувати архітектуру кластера;

– практично реалізувати СЗД;

– виконати інтеграцію з хмарною платформою OpenStack.

Об'єкт дослідження: хмарна платформа OpenStack зі сховищем Ceph.

Предмет дослідження: механізм інтеграції хмарної платформи OpenStack з кластером Ceph.

Наукова новизна отриманих результатів:

– докладно описано основні складові динамічного управління кластером (пули даних, присвоєння груп розміщення і призначення їх ID, ребалансування);

– відображено програмні особливості інтеграції хмарної платформи OpenStack з кластером Ceph (налаштування OpenStack для клієнтів та використання Ceph);

– програмно розроблено СЗД.

Практичне значення одержаних результатів. Впровадження розробки дозволить вигідно замінити дорогі сучасні системи зберігання даних, при цьому не поступаючись їм в продуктивності.

Апробація результатів. Окремі результати дослідження представлено на IX Міжнародній науково-технічній конференції молодих учених та студентів «Актуальні задачі сучасних технологій» Тернопільського національного технічного університету імені Івана Пулюя (25-26 листопада 2020 р.) у вигляді опублікованих тез [4].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вимоги до системи, яка розробляється

РСЗД призначена для зберігання даних, доступних для хмарної платформі OpenStack. Під даними в контексті хмарної платформи будемо вважати: [3, 4]

– ефемерні диски серверів (ephemeral Nova volumes);

– постійні диски (Cinder volumes);

– образи віртуальних серверів (Glance images).

В результаті створення даної системи зберігання даних повинні бути забезпечені наступні показники СЗД:

– масштабованість;

– відмовостійкість.

В системі функціонально визначені такі підсистеми:

– розподілене сховище даних;

– інтерфейси управління СЗД та хмарної платформою OpenStack.

Забезпечення пристосовності системи повинно виконуватися за рахунок:

– розширення сумарної ємності системи при додаванні нового сервера зберігання;

– зменшення сумарної ємності системи при видаленні сервера зберігання (наприклад, в результаті збою сервера);

– своєчасної заміни обладнання, яке вийшло з ладу;

– модернізації архітектури і інтерфейсу відповідно до нових вимогами;

– своєчасного адміністрування сервера;

– оперативного реагування на побажання користувачів.

Надійність СГД повинна забезпечуватися за рахунок:

– відсутності єдиної точки відмови в архітектурі;

– розподіленого зберігання даних з автоматичною репликацією по мережі між серверами;

– можливість відновлення необхідного рівня відмовостійкості при виході з ладу частини обладнання;

– можливості зміни рівня відмовостійкості (фактора реплікації) для різних пулів.

Система повинна надавати інтерфейс командного рядка для управління.

Інформаційна безпека в системі повинна здійснюватися за рахунок:

– доступу до управління по захищених протоколах SSH / HTTPS;

– аутентифікації СГД всередині кластера (між серверами).

Вимоги до розподіленого сховища інформації :

– використання тільки технологій з відкритими вхідними кодами;

– список підтримуваних операційних систем: Red Hat Enterprise Linux, CentOS, Fedora;

– автоматична реплікація даних між серверами;

– можливість відновлення даних після збою частини обладнання;

– можливість задання особливих параметрів реплікації даних з урахуванням можливих failure доменів

– нативна інтеграція з OpenStack (nova, cinder, glance і swift);

– можливість використання моментальних знімків (снепшот);

– можливість використання твердотільних накопичувачів для більш вимогливих до продуктивності системи зберігання клієнтів.

Вимоги до інтерфейсу управління сховищем даних:

– надання інформації за запитом;

– забезпечення цілісності інформації;

– забезпечення коректного відображення інформації.

Вимоги до ПЗ:

– Операційна система: система повинна бути спроектована для роботи під керуванням Red Hat Enterprise Linux 7 або аналогічних ОС (Fedora Linux 20, CentOS 7);

– СУБД: як БД (якщо потрібно для Web Інтерфейсу) має

використовуватися СУБД MySQL або MariaDB.

Для створення системи необхідно провести наступні види робіт:

– аналіз доступних Linux / Unix OpenSource РСЗД (включаючи розподілені файлові системи) на прикладі Ceph, GlusterFS, LustreFS, Swift, pNFS;

– вибір тієї РСЗД, яка найбільш повно задовольняє всім критеріям;

– вибір найкращих сценаріїв використання обраних СЗД;

– проектування архітектури;

– оптимізація продуктивності;

– інтеграція з хмарною платформою OpenStack.

1.2 Огляд платформи OPENSTACK

Проект OpenStack - це хмарна платформа з відкритим вихідним кодом. Він був заснований в 2010 році Rackspace і NASA для допомоги організаціям у використанні хмарних обчислень. Відмінними рисами даного проекту є простота реалізації, масштабованість і великий набір функцій. Над проектом працюють експерти в сфері хмарних обчислень з усього світу. [5,6]

OpenStack надає рішення Інфраструктура-як-сервіс (IaaS) при допомозі різних програмних компонентів. Кожен компонент пропонує API, що полегшує їх інтеграцію.

Ключові особливості OpenStack:

– швидке розгортання віртуальних серверів;

– мережа, яка програмно визначається (Software-defined network - SDN);

– єдина точка управління інфраструктурою;

– малі витрати зусиль для підтримки інфраструктури.

Для яких сценаріїв підходить OpenStack:

– хмарне навантаження: для створення і управління безліччю віртуальних машин з відносно коротким життєвим циклом (підхід свиноферми);

– середовища розробки та тестування;

– створення інфраструктури для Big data (Apache Hadoop або Apache Sparc).

Для чого не підходить OpenStack:

– заміна класичної корпоративної платформи віртуалізації;

– хостинг віртуальних серверів з великим життєвим циклом, які повинні бути захищені засобами підвищення відмовостійкості.

Разом з тим в OpenStack присутні наступні обмеження:

– відсутні кошти забезпечення високої доступності віртуальних серверів;

– обмежені операції з віртуальними машинами (загальні для всіх хмар);

– додаткове апаратне забезпечення (NIC, PCI пристрої) НЕ може бути додано через портал самоврядування (Dashboard), тільки через консоль;

– зміна розміру віртуальних машин вимагає великих трудовитрат.

Нижче буде наведено список сервісів OpenStack. [7]

Портал самообслуговування (Dashboard), проект Horizon. Надає веб-портал самообслуговування для взаємодії з основними сервісами OpenStack. Дозволяє виконувати такі операції, як запуск віртуального сервера, привласнення публічної IP адреси і налаштування доступу

Обчислення (Compute), проект Nova. Управляє життєвим циклом віртуальних серверів в середовищі OpenStack. Обов'язками сервісу є призупинення, планування та виведення з експлуатації віртуальних серверів на вимогу.

Мережа (Networking), проект Neutron. Надає підключення до мережі як послугу для сервісів OpenStack, таких як OpenStack Compute. Надає API користувачам для визначення мереж і приєднання до них. Володіє змінною архітектурою, яка підтримує безліч популярних мережевих вендорів і технологій.

Об'єктне сховище (Object Storage), проект Swift. Зберігає і добуває довільні об'єкти неструктурованих даних за допомогою RESTful, заснованому на HTTP API. Він високо відмовостійкий завдяки реплікації даних і архітектури, яка масштабується.

Блочне сховище (Block Storage), проект Cinder. Забезпечує блочне сховище для запуску віртуальних серверів. Його архітектура драйверів , що замінюються, полегшує створення і управління блоковими пристроями.

Сервіс ідентифікаціїї (Identity service), проект Keystone. Надає сервіс аутентифікації і авторизації для інших сервісів OpenStack. Також надає каталог endpoints для всіх сервісів OpenStack.

Сервіс образів (Image service), проект Glance. Зберігає і добуває образи дисків віртуальних машин. OpenStack Compute використовує цей сервіс під час створення віртуальних машин.

Сервіс обліку використання ресурсів (Telemetry), проект Ceilometer. Спостерігає за хмарою OpenStack і проводить виміри для виставлення рахунків, тесту продуктивності, масштабованості і статичних цілей.

Сервіс оркестрації (Orchestration), проект Heat. Організовує кілька складових хмарних додатків, використовуючи нативний шаблонний формат HOT або шаблонний формат AWSCloudFormation, за допомогою OpenStack-native REST API і CloudFormation-сумісний Query API.

Сервіс БД (Database service), проект Trove. Забезпечує масштабований і надійний хмарний БД-як-сервіс функціонал для реляційних і нереляційних СКБД.

Загальна архітектура платформи OpenStack відображена на рисунку 1.1.

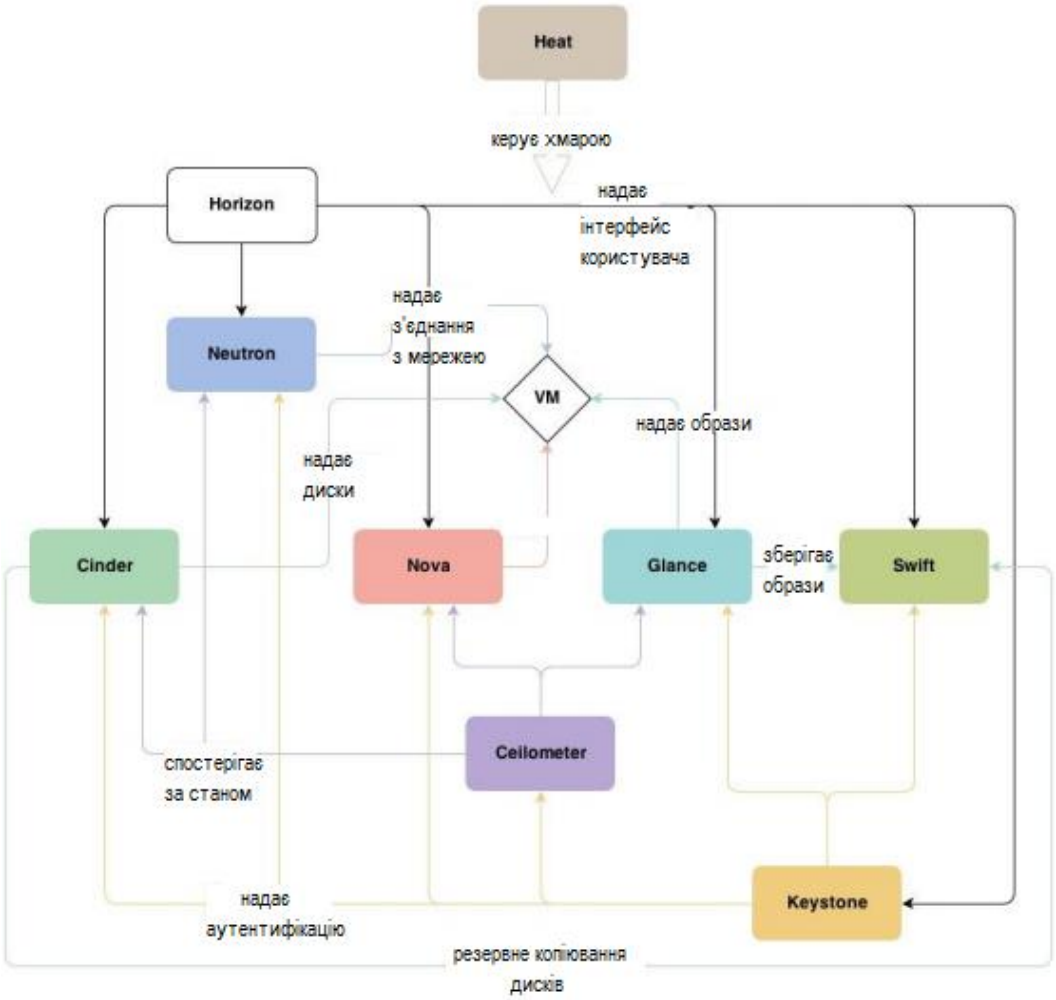

Рисунок 1.1 - Загальна архітектура платформи OpenStack

1.3 Огляд основних розподілених ФС

Метою цього порівняння є вибір найбільш оптимального рішення в області зберігання даних. Рішення повинно задовольняти вимогам комплексу ПЗ «OpenStack» [8].

В ході досліджень автором було проведено порівняння наступних РСЗД: LustreFS (v 2.0); pNFS (NFS 4.1); GlusterFS; Ceph; Swift.

Порівняння РСЗД проводилося виходячи з наступних критеріїв:

- масштабованість рішення;
- відмовостійкість рішення і методи догляду від «split brain»;
- оперативні витрати на управління рішенням (OpPEX);
- сумісність рішення з ПЗ «Openstack» (включаючи компоненти);

- швидкість I / O в послідовних операціях (читання і запис);
- швидкість I / O в випадкових операціях (читання і запис);
- підтримка реплікації (снепшот);
- підтримка можливостей журналювання на SSD дисках;
- POSIX сумісність;
- підтримка квот;
- підтримка на рівні ядра;
- поширеність і активність розробки;
- початкові витрати на рішення.

Вся інформація, необхідна про проведення порівняння, бралася з офіційної документації [11, 12].

1.3.1 LustreFS

Вона є типовою розподіленою ФС (Stripped FS) з виділеним сервером метаданих [13]. Вона оптимізована для дуже великих кластерів (~ 10 000 вузлів). Реплікацію даних і снепшот не підтримує. Доступ до даних здійснюється через клієнт Lustre. На даний момент вважається застарілою і слабо підтримується.

ФС LustreFS складається з трьох основних функціональних модулів:

– сервер метаданих (metadata server - MDS), який містить метадані про простори імен (в т.ч. імена файлів, каталогів, права доступу, карту розміщення файлів);

– сервери зберігання (object storage server - OSS), що зберігають дані файлів;

– клієнти, що використовують ці дані.

Lustre надає всім клієнтам уніфікованиий простір імен для всіх файлів і даних у ФС, використовуючи стандартну семантику POSIX, а також забезпечує паралельний доступ до запису та читання файлів в ФС.

1.3.2 pNFS

Забезпечує можливість паралельного читання і запису файлів з багатьох вузлів [14]. Використовує виділений сервер метаданих, до якого звертаються NFS клієнти. Підтримує як розподілений (distributed), так і striped режими. Доступ до даних здійснюється через клієнт NFS. Open source версія не має «Release» статусу, комерційні версії несумісні між собою.

РСЗД pNFS складається з трьох основних функціональних модулів:

– NFS сервер метаданих, який зберігає метадані про простори імен (в т.ч. імена файлів, каталогів, права доступу, а також карту розміщення файлів);

– сервери зберігання даних (файлів);

– клієнти NFS, які отримують інформацію від сервера NFS і які звертаються до серверів зберігання.

1.3.3 GlusterFS

Працює в призначеному для користувача просторі за допомогою технології FUSE і не має виділеного сервера метаданих [15]. GlusterFS може об'єднати сховища даних (на існуючих ФС ext4, XFS), що знаходяться на різних серверах, в одну паралельну мережеву ФС. РСЗД GlusterFS містить тільки один функціональний модуль - це вузли зберігання (Storage Node, brick).

На кожному такому вузлі працює служба glusterd, яка надає доступ клієнтам до локальних ФС. Таких brick-ів на одному сервері може бути безліч. Наприклад, на одному диску або масиві може бути кілька розділів з різними ФС, кожна з яких буде окремим brick-ми. Кілька серверів об'єднуються в кластер або пул зберігання даних (Trusted Storage Pool) в термінології GlusterFS. В рамках пулу зберігання підтому на різних вузлах об'єднуються в логічні томи (volumes) різної конфігурації. Основна відмінність від попередніх ФС в тому, що метадані зберігаються не на окремому сервері, а в додаткових атрибутах файлів в inode.

1.3.4 Swift

Надає розподілене об'єктне сховище, яке масштабується [16]. Об'єктне сховище не має концепції ФС, натомість кожен файл + метадані зберігаються як об'єкт. Відповідно, монтування такої системи зберігання даних (як частини локального дерева Linux) неможливо і команди vi або ls працювати не будуть. Доступ до об'єктів в Swift здійснюється по інтерфейсу REST. Ці об'єкти можна зберігати, отримувати або оновлювати за вимогою.

Основна відмінність об'єктного сховища від традиційної ФС - об'єкти (файли і т.п.) можуть завантажуватися або оновлюватися цілком. Немає можливості оновити тільки частину даних в розглянутому об'єкті.

Існує 4 основних компоненти Swift:

- Proxy Server (проксі-сервер): об'єднує всі компоненти системи разом;
- Object Server (об'єктний сервер): відповідальний за зберігання даних;
- Container Server (контейнерний сервер): в його функції входить віддача списку об'єктів;
- Account Server (сервер аккаунтинга): віддає списки вмісту контейнерів для конкретного аккаунта.

Типова Swift-інфраструктура являє собою кластер, одна з машин якого працює як проксі-сервер, кілька машин працюють в якості контейнерних серверів і серверів акаунтингу, а інші є контейнерними серверами.

1.3.5 Ceph

Є універсальною, тобто може працювати як розподілена ФС, блокова розподілена система і як об'єктне сховище [17, 18]. Це досягається завдяки розвитку системи через взаємну інтеграцію трьох інтерфейсів:

– RADOS Gateway - S3- і Swift-сумісний RESTful інтерфейс;
– RBD - блоковий пристрій з підтримкою тонкого зростання і снепшот;
– CephFS - розподілена POSIX-сумісна ФС;

РСЗД Ceph містить наступні основні функціональні модулі:

– Metadata server (MDS) - демон для забезпечення синхронного стану файлів в точках монтування CephFS. Працює за схемою активна копія + резерви, причому активна копія в межах кластера тільки одна. Даний демон необхідний тільки при використанні CephFS;

– Mon (Monitor) - елемент інфраструктури Ceph, що забезпечує адресацію даних усередині кластера і зберігає інформацію про топології, стани і розподіли даних усередині сховища. Клієнт, який бажає звернутися до блокового пристрою rbd або до файлу на домонтованій cephfs, отримує від монітора ім'я і положення rbd header -спеціального об'єкта, що описує становище інших об'єктів, що відносяться до запитаної абстракції (блоковий пристрій або файл) і далі спілкується з усіма OSD, які беруть участь в зберіганні файлу;

– OSD (object storage daemon) - демон, який відповідає за зберігання даних, основний будівельний елемент кластера Ceph. На одному фізичному сервері може розміщуватися кілька OSD, кожен з яких має під собою окреме фізичне сховище даних (окремий диск);

– об'єкт (Object) - блок даних фіксованого розміру;

– група розміщення (Placement Group, PG) - логічна група, яка об'єднує безліч об'єктів, призначена для спрощення адресації і синхронізації об'єктів. Окремо кожен об'єкт існує лише в одній плейсмент групі;

– пул (pool) - об'єднання PG. Глобальні налаштування беруться для пулу.

Підтримує режими роботи з чергуванням (stripped) і реплікацією та режим з шифруванням (erasure coded). В основі лежить об'єднання дискових просторів декількох (десятків, сотень) серверів в об'єктне сховище, що дозволяє реалізувати гнучку багаторазову псевдовипадкову надмірність даних. Кожен OSD має власний журнал, який синхронізується з локальною ФС.

1.3.6 Порівняння основних параметрів розглянутих розподілених ФС

У таблиці 1.1 представлено підсумкове порівняння параметрів ФС на підставі даних з офіційної документації. Результати наведено за шкалою від 0

до 3, де 0 - не реалізовано зовсім, 3 – реалізовано повністю. Рейтинг – сума показників усіх оцінених параметрів для кожної ФС.

Таблиця 1.1 - Перелік перевірок і результатів випробувань

№з/п	Найменування параметру	LustreFS	pNFS	GlusterFS	Ceph	Swift
1.	Масштабованість рішення	2	1	2	3	3
2.	Відмовостійкість рішення і шляхи обходу «Split brain»	1	0	1	3	2
3.	Оперативні витрати на управління рішенням (OpPEX)	1	1	3	2	3
4.	Сумісність рішення з ПЗ «Openstack»	0	0	3	3	3
5.	Підтримка реплікації (Снепшот)	0	0	2	3	2
6.	Підтримка можливостей ведення журналів на SSD дисках	3	0	2	3	1
7.	POSIX сумісність	2	1	3	2	1
8.	Підтримка квот	2	0	2	2	3
9.	Підтримка на рівні ядра	0	2	3	3	3
10.	Поширеність і активність розробки	0	0	3	3	3
11.	Початкові витрати на рішення (CaPEX)	0	0	3	2	3
12.	Підтримка режиму кешування	0	0	1	3	0
13.	Рейтинг	11	5	28	32	27

За результатами порівняння для створення СЗД була обрана реалізація на базі Ceph в replication pool mode через RBD без CephFS [18, 19].

1.4 Висновки до першого розділу

У першому розділі проведено аналіз предметної області дослідження, зокрема сформульовані вимоги до розроблюваної РСЗД. Зроблено докладний опис хмарної платформи OpenStack. Проведено аналіз окремих СЗД. Порівняно їх основні функціональні можливості та параметри. За результатами порівняння для виконання поставленого завдання на проектування обрано середовище Ceph.

2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Архітектура Ceph

Ceph надає об'єктне, блочне і файлове сховища в одній уніфікованій системі. Він високонадійний, безкоштовний і легко керований. Ceph надає широкі можливості масштабування - до декількох тисяч клієнтів, що працюють з пета- і ексабайт даних. Вузли Ceph використовують стандартне обладнання та сервіси, а кластер зберігання даних Ceph вміщує велику кількість вузлів, які взаємодіють один з одним для реплікації і динамічного перерозподілу даних. [18]

Операційні системи. Нативні клієнти Ceph є тільки для операційних систем Linux, в операційних системах Windows можливе використання у вигляді диска, наданого S3-сумісним додатками.

Інтерфейси управління. Ceph підтримує такі інтерфейси управління: інтерфейс командного рядка (CLI), API для Ci (або C ++), API для Python, restAPI, веб інтерфейси (Intel VSM, Calamari).

Структура. Кластер Ceph складається з демонів трьох типів:

– Ceph монітор ;
– Сервер метаданих Ceph (MDS);
– Ceph OSD демони.

Ceph монітор має в своєму розпорядженні основний копією карти кластера. Кластер з Ceph моніторів забезпечує високу доступність даних в разі падіння одного з демонів моніторів. Клієнти отримують копію карти кластера від Ceph монітора.

Ceph MDS зберігає всі метадані ФС.

Ceph OSD демон періодично перевіряє свій статус і статус інших

OSD демонів і відправляє звіти моніторів.

Клієнти Ceph. Клієнти Ceph включають в себе ряд інтерфейсів сервісів:

– блокові пристрої: Сервіс блокових пристроїв Ceph (також відомий як RBD) надає змінювані в розмірі, тонкі блокові пристрою з підтримкою снепшот і клонування. Ceph чергує блокові пристрої в кластері для високої продуктивності. Ceph підтримує і об'єкти ядра, і QEMU гипервізор, який використовує librbd безпосередньо – в обхід непродуктивних витрат об'єктів ядра для віртуальних систем;

– об'єктне сховище: Сервіс об'єктного сховища Ceph (також відомий як RGW) надає RESTful APIs з інтерфейсами, сумісними з Amazon S3 і OpenStack Swift;

– ФС: Сервіс ФС Ceph (CephFS) надає POSIX-сумісну ФС, яку можна використовувати за допомогою команди mount або як ФС в просторі користувача (filesytem in user space - FUSE). Ceph може запускати додаткові екземпляри OSD, MDS і моніторів для забезпечення масштабованості і високої доступності.

Рисунок 2.1 відображає високорівневу архітектуру.

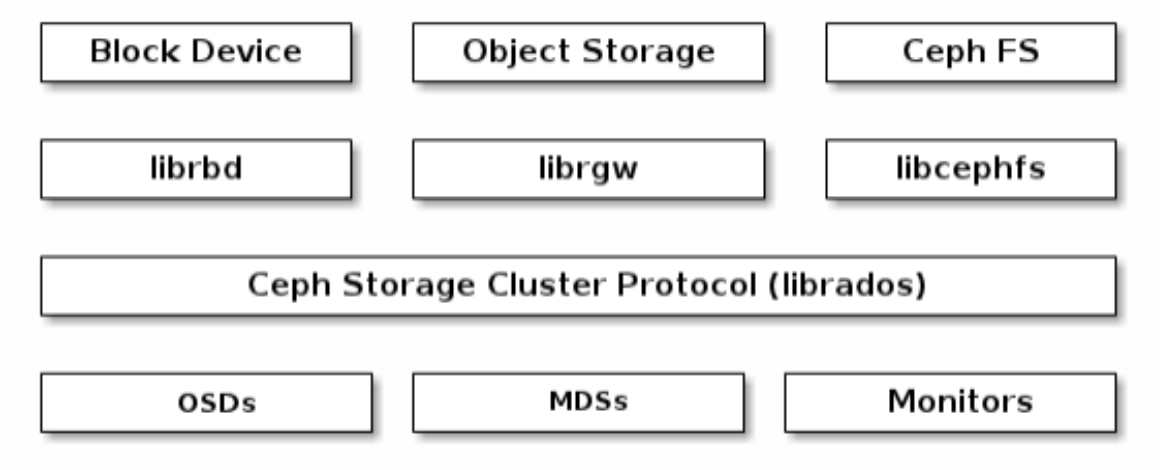

Рисунок 2.1 – Високорівнева архітектура Ceph

Об’єктне сховище Ceph. Демон об'єктного сховища Ceph, radosgw, це сервіс FastCGI, який надає RESTful HTTP API для зберігання об'єктів і метаданих. Це верхній шар кластера Ceph з його власними форматами даних, і він управляє своїми власними БД користувачів, аутентифікацією і контролем

доступу. RADOS Gateway використовує уніфікованих1 простір імен, що дозволяє використовувати як OpenStack Swift-сумісні API, так і Amazon S3-сумісні API.

Блоковий пристрій Ceph. Чергує образ блочного пристрою по багатьом об'єктах в кластері Ceph, де кожен об'єкт присвоюється групі розміщення, а групи розміщення поширені по різним ceph-osd демонам по всьому кластеру.

Чергування дозволяє блоковим пристроям RBD видавати більшу продуктивність, ніж у одного стандартного сервера.

Тонких1 блоковий пристрій Ceph з можливістю створення снепшотів є найбільш зручним варіантом для віртуалізації і хмарних обчислень. В сценаріях віртуальних машин, зазвичай, розгортається блоковий пристрій Ceph з rbd драйвером в QEMU / KVM, де хост використовує librbd для надання сервісу блочного пристрою гостю.

ФС Ceph. CephFS надає POSIX-сумісну ФС як сервіс, який є верхнім шаром заснованого на об'єктах кластера Ceph. Файли CephFS присвоюються об'єктам, які Ceph зберігає в кластері. Клієнти Ceph монтують ФС CephFS як об'єкт ядра або як FUSE.

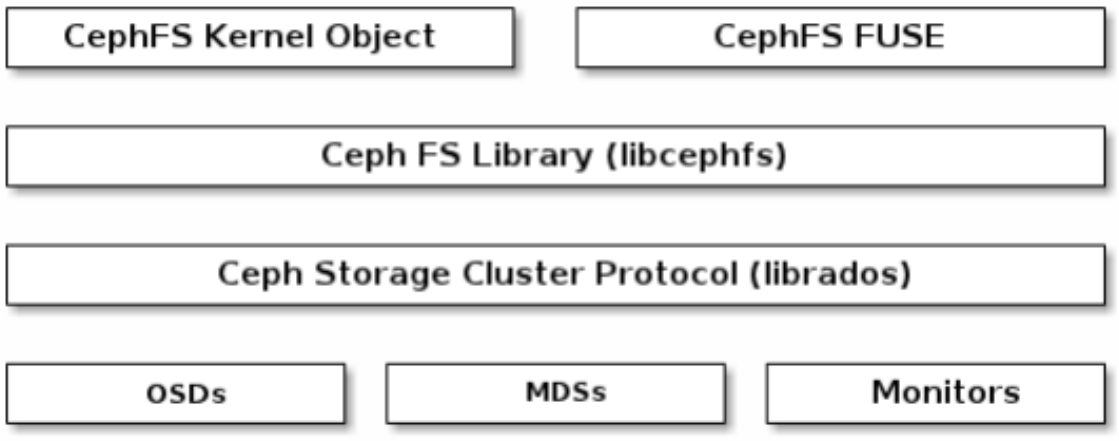

Рисунок 2.2 – Сервіс ФС Ceph

Сервіс ФС Ceph (рис. 2.2) включає в себе сервер метаданих Ceph (MDS), розгорнутий разом з кластером Ceph. Призначення MDS в зберіганні всіх

метаданих ФС (директорії, власники файлів, режим доступу, і т.д.) на високо доступних серверах метаданих Ceph, де метадані знаходяться в пам'яті. Причина для використання MDS (демон, який має назву ceph-mds) в тому, що прості операції з ФС, такі як вивід вмісту директорії або зміна директорії (ls, cd) будуть зайве займати ресурси Ceph OSD демонів. Таким чином, відділення метаданих від даних призводить до того, що ФС Ceph може надавати сервіси з високою продуктивністю, не навантажуючи кластер.

CephFS відокремлює метадані від даних, записуючи метадані в MDS і зберігаючи дані у файлі в одному або більше об'єктах кластера Ceph. ceph-mds може бути запущений як один процес, або він може бути розподілений на кілька фізичних машин, як для високої доступності, так і для масштабованості.

Висока доступність: Додаткові екземпляри ceph-mds можуть бути в запасі, готові до взяття операцій під свій контроль в випадку збою активного ceph-mds. Це легко, тому що вся інформація, включаючи журнал, зберігається на RADOS. Переміщення даних запускається автоматично завдяки ceph-mon.

Масштабованість: Безліч екземплярів ceph-mds можуть бути активні, і вони будуть ділити дерево директорій на піддерева (і частини однієї зайнятої директорії), ефективно розподіляючи навантаження між усіма активними серверами.

Комбінації активних і очікуючих сервісів можливі, наприклад, запуск трьох активних примірників ceph-mds для масштабованості і одного екземпляру в режимі очікування для високої доступності.

2.2 Зберігання даних

Кластер даних на Ceph отримує дані від клієнтів Ceph- нералежно від того, використовується при цьому блоковий пристрій Ceph , об'єктне сховище Ceph, ФС Ceph або клієнтська реалізація на основі librados - і зберігає дані як об'єкти [21]. Кожен об'єкт відповідає файлу в ФС, який зберігається в пристрої

зберігання об'єктів (Object Storage Device , OSD). Ceph OSD демони управляють операціями читання / запису на дисках (рисунок 2.4).

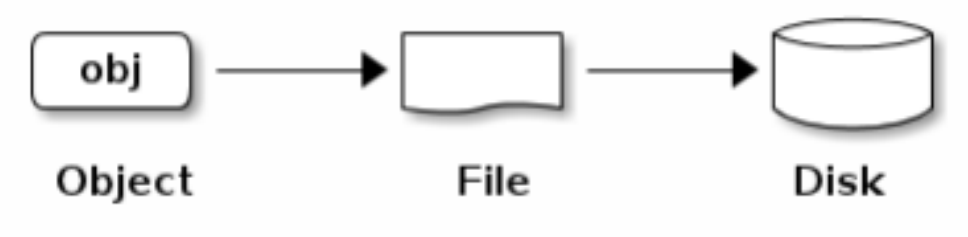

Рисунок 2.4 – Зберігання даних Ceph

Ceph OSD демони зберігають всі дані як об'єкти в плоскому просторі назв (без ієрархії директорій). У об'єкта є ідентифікатор (ID), двійкова інформація та метадані, що складаються з набору пар ім'я / значення. Семантика метаданих повністю залежить від типу доступу клієнтів Ceph . Наприклад, CephFS використовує метадані для зберігання атрибутів файлу: власник файлу, дата створення, дата останнього зміни і так далі. ID об'єкта є унікальним не тільки в рамках локальної ФС, але і в рамках всього кластера (рис. 2.5).

ID	Binary Data	Metadata
1234	01010101010101001101010010 01011000010101001101010010 01011000010101001101010010	name1 value1 name2 value2 nameN valueN

Рисунок 2.5 – Зберігання даних Ceph

2.3 Масштабованість і висока доступність

У традиційних архітектурах клієнти взаємодіють з централізованим компонентом (наприклад, шлюз, API і так далі), який діє як єдина точка входу в складну підсистему. Це призводить до обмеженням в продуктивності і масштабованості і являє собою єдину точку відмови (наприклад, якщо

централізований компонент перестане працювати, вся система перестане працювати теж) [21].

Ceph не має єдиного централізованого компонента, що дозволяє клієнтам взаємодіяти з Ceph OSD демонами безпосередньо. Ceph OSD демони створюють репліки об'єктів на інших нодах Ceph для збільшення збереження і доступності даних. Ceph також використовує кластер моніторів для забезпечення високої доступності. Для усунення централізованності Ceph використовує алгоритм під назвою CRUSH.

Ceph клієнти і Ceph OSD демони використовують CRUSH алгоритм для ефективного обчислення відповідного місця розташування об'єкта незалежно від якоїсь довідкової таблиці. CRUSH використовує інтелектуальну схему реплікації даних для забезпечення відмовостійкості, що найкращим чином підходить для лінійно масштабованих сховищ даних.

Карта кластера. Робота кластера Ceph залежить від того, чи знають Ceph клієнти і Ceph OSD демони топологію кластера, яка включає в себе 5 карт, які складають в сумі «Карту кластера»

1. Карта моніторів: Містить fsid кластера, позицію, ім'я, адреса і порт кожного монітора. Вона також визначає поточний період, дату створення карти і останнім часом зміни. Команда для перегляду карти моніторів ceph mon dump.

2. Карта OSD: Містить fsid кластера, дату створення і останньої модифікації карти, список пулів, кількість копій, кількість PG, список OSD і їх статус. Команда для перегляду карти OSD ceph osd dump.

3. Карта PG: Містить версію групи розміщення, час, поточний період карти OSD, співвідношення і деталі по кожній групі розміщення: ID, стан кожної групи розміщення і статистику використання даних для кожного пулу. Команда для перегляду карти груп розміщення ceph pg dump.

4. CRUSH карта: Містить список пристроїв, ієрархію доменів падіння (пристрій, хост, стійка, ряд, кімната і т.д.) і правила для переміщення по ієрархії при запису даних. Щоб подивитися CRUSH карту, необхідно виконати наступні

команди:

```
ceph osd getcrushmap -o {filename}
crushtool -d {comp-crushmap-filename} -o {decomp-crushmap-filename}
```

Перша команда видасть карту в нечитабельним вигляді, друга проведе декомпіляцію карти. Декомпільовану карту можна переглянути в текстовому редакторі або за допомогою команди cat.

5. Карта MDS: Містить поточний період MDS карти, дати створення і останньої зміни карти. Також містить пул для зберігання метаданих, список серверів метаданих та їх статуси. Команда для перегляду карти MDS ceph mds dump.

Кожна карта підтримує ітеративну історію змін її робочого стану. Монітори Ceph підтримують головну копію карти кластера, що включає в себе членів кластера, їх стан, зміни і загальний стан кластера Ceph.

Висока доступність моніторів. Перед тим, як клієнти Ceph зможуть читати або записувати дані, вони повинні зв'язатися з монітором Ceph для отримання останньої копії карти кластера. Кластер даних на Ceph може працювати з одним монітором, але це буде єдиною точкою відмови (якщо монітор стане недоступний, клієнти Ceph НЕ зможуть читати або записувати дані).

Для підвищення надійності та відмовостійкості Ceph підтримує кластер моніторів. У кластері моніторів затримка і інші збої можуть привести до того, що один або більше моніторів не матимуть інформацію про поточний стан кластера. З цієї причини в Ceph завжди повинна бути домовленість між різними моніторами щодо стану кластера. Ceph завжди використовує кворум моніторів (наприклад, 1, 2: 3, 3: 5, 4: 6, і т.д.) для досягнення консенсусу між моніторами про поточний стан кластера.

Висока доступність аутентифікації. Для ідентифікації користувачів і захисту проти атак «людина всередині» Ceph надає власну систему аутентифікації cephx для аутентифікації користувачів і демонів. Протокол cephx

не підтримує шифрування даних в транспорті (наприклад, SSL / TLS) або шифрування в REST.

Cephx використовує загальні секретні ключі для аутентифікації, такі, що і клієнт, і монітор мають копію клієнтського секретного ключа. Головною особливістю масштабованості Ceph є відсутність централізованого інтерфейсу об'єктного сховища Ceph, тому клієнти Ceph повинні бути здатні взаємодіяти з OSD безпосередньо.

Користувач викликає клієнта Ceph для зв'язку з монітором. Кожен монітор може провести аутентифікацію користувачів і поширити ключі всередині кластера. Монітор повертає структуровані дані аутентифікації, які містять ключ сесії для використання при підключенні до сервісів Ceph. Ключ сесії зашифрований постійним секретним ключем користувача. Потім клієнт використовує ключ сесії для запиту необхідних сервісів від монітора, і монітор надає клієнту тікет, який буде використаний при аутентифікації користувача на OSD. Монітори Ceph і OSD поширюють ключ між собою, що дозволяє клієнту використовувати тікет, наданий монітором, з будь-яким демоном OSD або сервером метаданих в кластері. Cephx тікети мають термін придатності. Така форма аутентифікації допомагає запобігти атакам з доступом до середовища передачі даних.

Для використання cephx адміністратор повинен налаштувати користувачів. В наступній діаграмі (рис. 2.6) client.admin користувач викликає ceph auth get-or-create-key з командного рядка для генерації імені користувача і секретного ключа. Підсистема Ceph auth генерує ім'я користувача і ключ, копіює їх на монітори і передає дані користувачу client.admin.

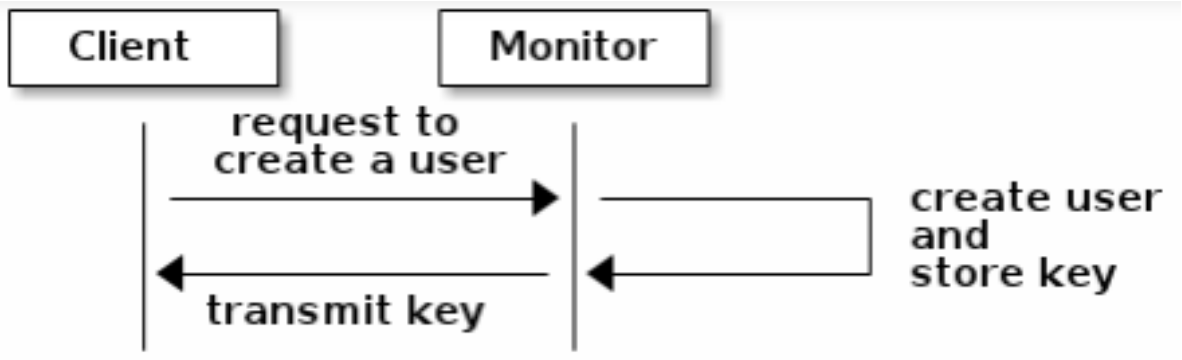

Рисунок 2.6 – Налаштування користувачів Ceph

Для аутентифікації моніторами клієнт відправляє ім'я користувача,монітор генерує ключ сесії і шифрує його за допомогою секретного ключа, належить користувачеві (рис. 2.7).

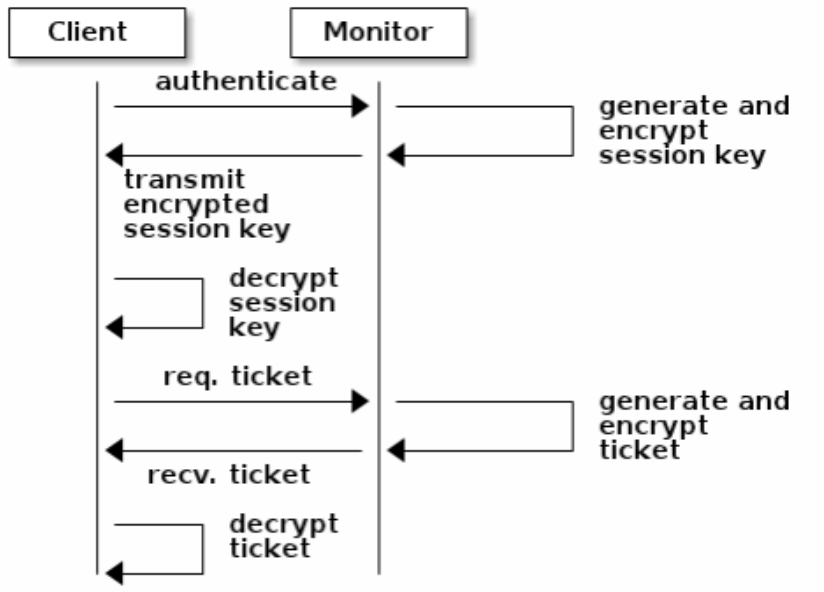

Рисунок 2.7 – Аутентифікація моніторами (частина 1)

Потім монітор відправляє зашифрований тікет клієнта. Клієнт дешифрує його за допомогою секретного ключа для отримання ключа сесії. Ключ сесії потрібен для ідентифікації користувача в поточній сесії. Після цього клієнт запитує тікет від імені користувача, певного ключем сесії. Монітор генерує тікет, шифрує його за допомогою секретного ключа користувача і відправляє

клієнту. Клієнт дешифрує тікет і використовує його для відправки запитів демонам OSD і серверів метаданих по всьому кластеру (рис. 2.8).

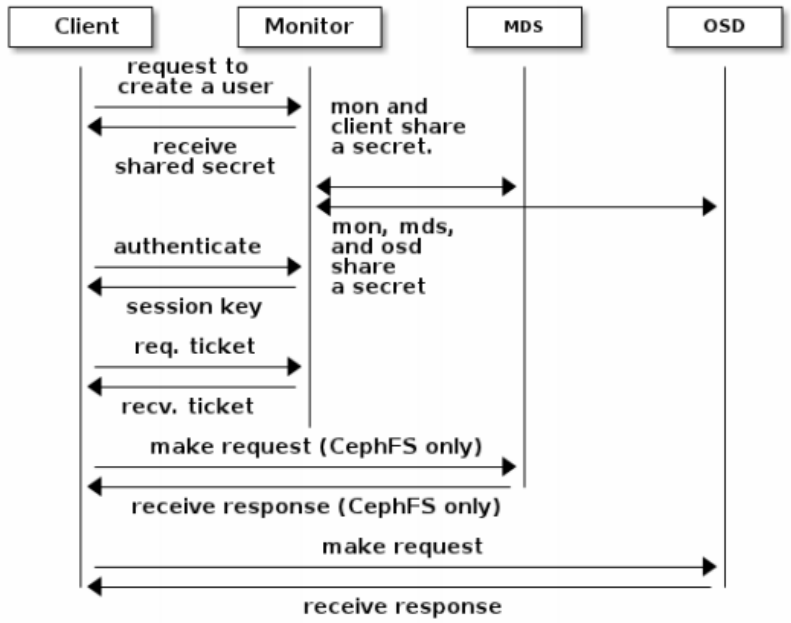

Рисунок 2.8 – Аутентифікація моніторами (частина 2)

Захист, пропонований цією аутентифікацією, існує тільки між клієнтом Ceph і хостами сервера Ceph. Не існує аутентифікації між клієнтами Ceph. Якщо користувач підключається до клієнта Ceph з віддаленого хоста, Ceph аутентифікація не гарантує безпеку з'єднання між призначеним для користувача хостом і хостом клієнта.

Гіпермасштабованість**.** У багатьох кластерних архітектурах основний сенс членства в кластері в тому, щоб централізованих інтерфейс знав, до яких вузлів він має доступ. Потім централізований інтерфейс надає сервіси клієнту через подвійну маршрутизацію, що є вузьким місцем для сховищ петабайтних-ексабайтних розмірів.

Ceph не має цього вузького місця: Ceph OSD демони і клієнти знають структуру кластера. Як і клієнти Ceph, кожен Ceph OSD демон знає про інших Ceph OSD демонів в кластері. Це дозволяє Ceph OSD демонам взаємодіяти

безпосередньо один з одним і з моніторами Ceph. На додачу, це дозволяє клієнтам Ceph безпосередньо взаємодіяти з Ceph OSD демонами.

Здатність клієнтів Ceph, моніторів Ceph і Ceph OSD демонів взаємодіяти один з одним означає, що Ceph OSD демони можуть використовувати CPU і RAM вузол Ceph для виконання завдань замість навантаження централізованого сервера.

Можливість використовувати цю обчислювальну потужність призводить до виникнення таких переваг.

1. Обслуговування клієнтів безпосередньо OSD: Оскільки будь-який мережевий пристрій має обмеження за кількістю одноразових підключень, централізована система має низький фізичний ліміт при використанні кластерів великого масштабу. Дозволяючи клієнтам Ceph взаємодіяти з Ceph OSD демонами безпосередньо, Ceph збільшує одночасно і продуктивність, і загальну пропускну здатність системи, видаляючи при цьому єдину точку відмови. Клієнти Ceph можуть зберігати сеанс з конкретним Ceph OSD демоном замість сеансу з централізованим сервером.

2. Членство і статус OSD: Ceph OSD демони приєднуються до кластеру і повідомляють про свій статус. На найнижчому рівні статус Ceph OSD демона up або down відображають, чи працює він і чи здатний обслуговувати запити клієнтів Ceph. Якщо Ceph OSD демон down і in в кластері зберігання даних Ceph, цей статус може означати падіння Ceph OSD демона. Якщо Ceph OSD чорт не запущений (наприклад, виник збій при запуску), Ceph OSD демон не може повідомити монітор Ceph про своє down статус. Монітор Ceph може періодично звертатися до Ceph OSD демона, щоб переконатися в тому, що він працює. Ceph також уповноважує всі Ceph OSD демони визначати стан сусідніх OSD, оновлювати карту кластера і повідомляти про це монітори Ceph. Таким чином, монітору Ceph дістаються процеси, які мають легку вагу.

3. Очищення даних: В рамках підтримування цілісності і чистоти даних Ceph OSD демони можуть видаляти об'єкти з груп розміщення. Ceph OSD

демони можуть порівнювати метадані об'єкта в одній групі розміщення з репліками в групах розміщення, що зберігаються на інших OSD. Очищення (зазвичай проводиться щодня) відловлює баги і помилки ФС. Ceph OSD демони також виконують більш глибоке очищення, порівнюючи дані в об'єктах по-бітово. Глибоке очищення (зазвичай проводитьсщотижня) знаходить пошкоджені сектораи на диску, які були пропущені при легкому очищенню.

4. Реплікації: Як і клієнти Ceph, Ceph OSD демони використовують CRUSH алгоритм, але Ceph OSD демони використовують його для визначення, де повинні зберігатися репліки об'єктів (і для ребалансування). У типовому сценарії запису клієнт використовує CRUSH алгоритм для визначення місця запису об'єкта, присвоєння об'єкту пулу і групі розміщення і визначення основного OSD для групи розміщення.

Клієнт записує об'єкт в групу розміщення на основний OSD. Потім основний OSD по своїй власній копії CRUSH карти визначає вторинний і третинний OSD (стільки OSD, скільки потрібно додаткових реплік), виконує реплікацію о'єкта до відповідних груп розміщення на вторинний і третинний OSD і відповідає клієнтові, підтверджуючи, що об'єкт був успішно записаний (рис. 2.9)

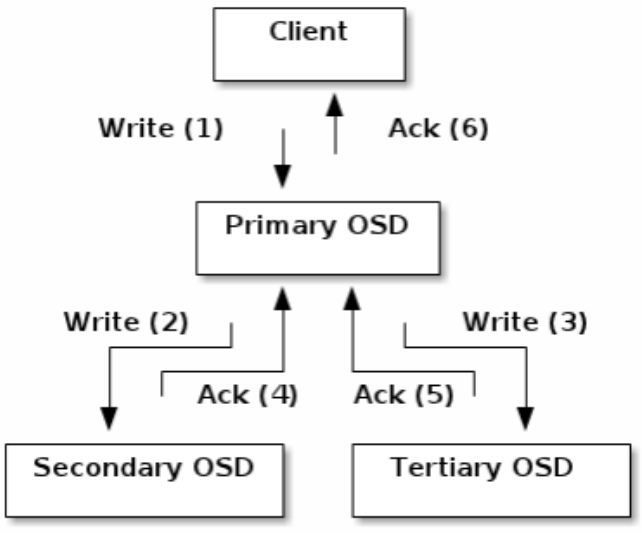

Рисунок 2.9 – Реплікація об’єктів в Ceph

Завдяки здатності виконувати реплікацію даних, Ceph OSD демони звільняють клієнтів Ceph від цього обов'язку, при цьому забезпечуючи високу доступність і безпеку даних.

2.4 Динамічне управління кластером

Пули даних. Система зберігання Ceph підтримує поняття «Пули», які є логічними розділами для зберігання об'єктів.

Клієнти Ceph отримують карту кластера від монітора Ceph і записують об'єкти в пули (рис. 2.10). Параметр пулу size або кількість реплік, набір правил CRUSH і кількість груп розміщення визначають, як Ceph розташує дані [22].

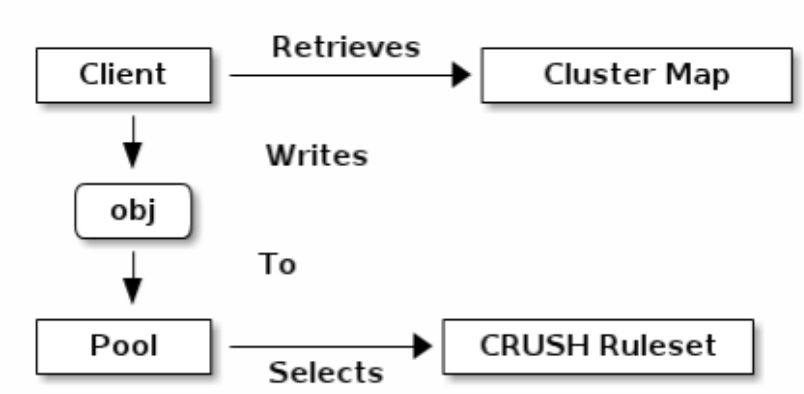

Рисунок 2.10 – Пули даних

Пули встановлюють такі параметри:

– володіння / доступ до об'єктів;
– кількість груп розміщення;
– набір правил CRUSH.

Присвоєння груп розміщення OSD. Кожен пул має деяку кількість груп розміщення. CRUSH присвоює групу розміщення OSD динамічно. Коли клієнт Ceph записує об'єкти, алгоритм CRUSH привласнює кожен об'єкт групі розміщення.

Присвоєння об'єктів груп розміщення створює проміжний шар між Ceph OSD демоном і клієнтом Ceph. Кластер зберігання даних Ceph повинен бути здатний рости (або розширюватися) і проводити динамічне ребалансування місць зберігання об'єктів. Якщо клієнт Ceph «знає», який Ceph OSD демон який об'єкт зберігає, це створить жорсткий зв'язок між клієнтом Ceph і Ceph OSD демоном. Замість цього, CRUSH алгоритм присвоює кожен об'єкт групі розміщення і потім привласнює кожну групу розміщення одному або більше Ceph OSD демонам. Цей проміжний рівень дозволяє Ceph проводити ребалансування динамічно, коли новий Ceph OSD демон і стоять за ним пристрою OSD з'являються в мережі.

Рисунок 2.11 зображує, як CRUSH присвоює об'єкти групам розміщення, і групи розміщення - OSD.

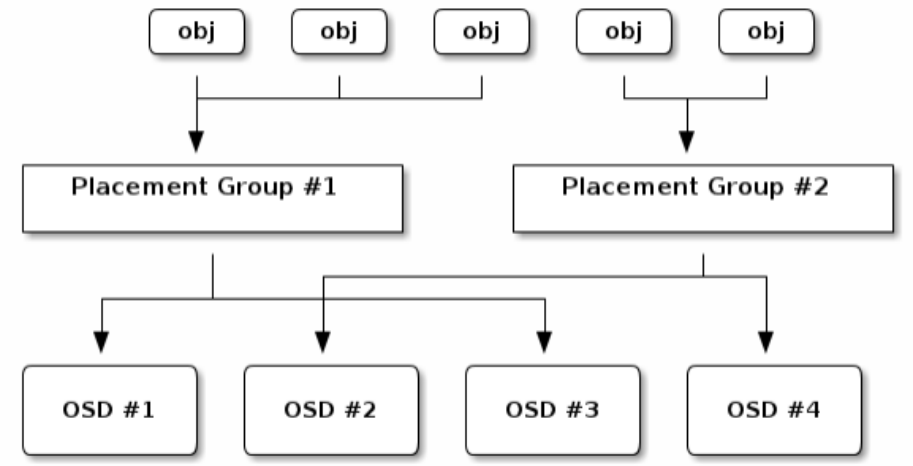

Рисунок 2.11 – Схема процесу присвоєння CRUSH

З копією карти кластера і CRUSH алгоритмом клієнт може обчислити, який саме OSD використовувати для читання або запису конкретного об'єкта.

Обчислення ID групи розміщення. Коли клієнт Ceph звертається до монітора Ceph, він отримує найбільшу останню копію карти кластера. З картою

кластера клієнт знає все про монітори, OSD і сервери метаданих в кластері. Проте, він не знає нічого про розташування об'єктів.

Розташування об'єктів обчислюється.

Єдині вхідні дані, необхідні клієнту, це ID об'єкта і пул. Ceph зберігає дані в іменованих пулах (наприклад, "liverpool"). Коли клієнт хоче записати іменований об'єкт (наприклад, "john", "paul", "george", "Ringo", і т.д.), він обчислює групу розміщення, використовуючи ім'я об'єкта, хеш-код, кількість груп розміщення в пулі і ім'я пулу. Клієнти Ceph використовують наступні кроки для обчислення ID групи розміщення:

1. Клієнт віддає ID пулу і ID об'єкта (наприклад, pool = "liverpool" і object-id = "john")

2. Ceph приймає ID об'єкта і хешує його.

3. Ceph обчислює хеш-модуль кількості груп розміщення (наприклад, 58), щоб отримати ID групи розміщення.

4. Ceph отримує ID пулу по його імені (наприклад, "liverpool" = 4)

5. Ceph додає ID пулу до ID групи розміщення (наприклад, 4.58).

Обчислення розташування об'єктів набагато швидше, ніж виконання узгодження місця розміщення об'єктів в режимі діалогу. CRUSH алгоритм дозволяє клієнту обчислювати, де повинен зберігатися об'єкт, і взаємодіяти з основною OSD для запису або отримання об'єктів.

Пірінг і групи. У попередньому розділі було відзначено , що Ceph OSD демони перевіряють стани один одного і повідомляють про це монітор Ceph. Ще одне, що Ceph OSD демони вміють робити, це пірінг (peering). Він являє собою процес приведення всіх OSD, які зберігають групу розміщення (PG), в згоду щодо стану всіх об'єктів (і їх метаданих) в цій PG.

Кластер зберігання даних Ceph був спроектований для зберігання принаймні двох копій об'єкта (наприклад, size = 2), що є мінімальними вимогами для збереження даних. Для високої доступності кластер даних Ceph повинен зберігати більше двох копій об'єкта (наприклад, size = 3 і min size = 2),

це може привести до появи статусу degraded в процесі забезпечення схоронності даних.

За угодою, Основний OSD це перший OSD в чинному наборі (Acting Set), і він відповідає за координацію процесу пірінга для кожної групи розміщення і приймає запити від клієнта на запис об'єктів в надану групу розміщення, де він бере участь як Основний.

Якщо ряд OSD є відповідальним за групу розміщення, тоді цей ряд OSD називається Acting Set. Acting Set може посилатися на Ceph OSD демонів, відповідальних за групу розміщення.

Ceph OSD демони, які є частиною Acting Set, можуть не завжди бути up. Коли OSD в Acting Set up, він також є частиною робочого набору (Up Set). Між Acting Set і Up Set велика різниця, тому що Ceph може привласнювати PG іншим Ceph OSD демонам, коли OSD перестає працювати.

У Acting Set для групи розміщення, що містить osd.25, osd.32 і osd.61, перший OSD, osd.25, є Основним. Якщо цей OSD перестане працювати, Вторинний osd.32 стане Основним, і osd.25 буде видалений з Up Set.

Ребалансування. Коли відбувається додавання OSD демона до кластеру зберігання даних Ceph, карта кластера оновлюється з новим OSD. Отже, змінюється розташування об'єктів, тому що змінюються вхідні дані для обчислень. Діаграма на рис. 2.12 зображує процес ребалансування, в якому деякі групи розташування мігрують з існуючих OSD (OSD 1 і OSD 2) на новий OSD (OSD 3). Навіть під час ребалансування CRUSH стабільний.

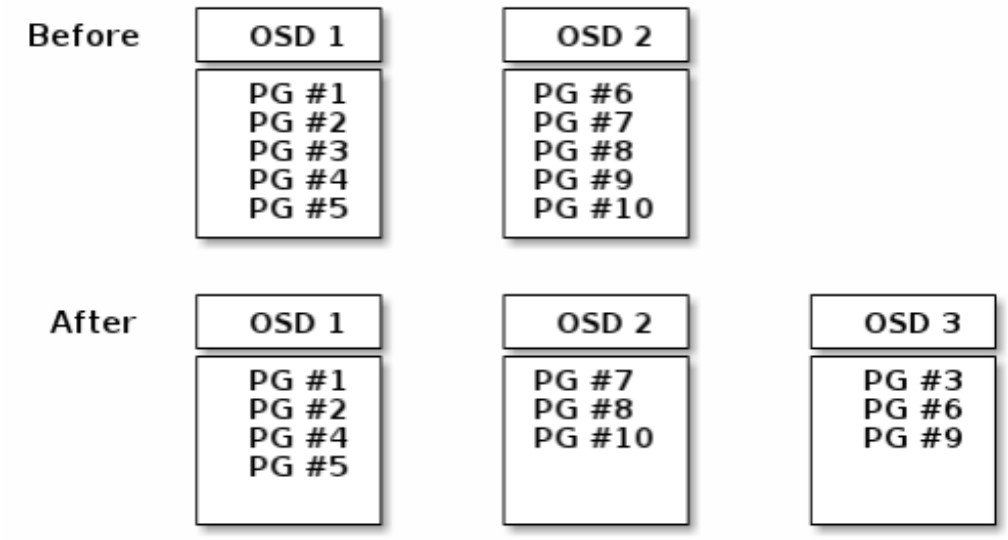

Рисунок 2.12 – Процес ребалансування

2.5 Чергування даних

Пристрої зберігання мають обмеження пропускної здатності, що впливає на продуктивність і масштабованість. Тому системи зберігання даних часто підтримують чергування - зберігання різних частин інформації на різних пристроях - для збільшення пропускної здатності і продуктивності. Основна форма чергування даних – це RAID. Найбільш близький до Ceph рівень RAID це RAID 0, або «тому з чергуванням». Чергування Ceph пропонує пропускну здатність RAID 0 і надійність n-кратного RAID зеркалення [23].

Ceph надає три типи клієнтів: блокові пристрої Ceph, ФС Ceph і об'єктне сховище Ceph. Клієнт Ceph конвертує свої дані з формату представлення, в якому їх бачить користувач (блоковий пристрій, RESTful об'єкти, директорії файлової системи CephFS) в об'єкти для пристрою зберігання даних Ceph.

Найпростіший формат чергування Ceph включає в себе чергування розміром в 1 об'єкт (рис. 2.13). Клієнти Ceph записують чергуються куски даних - страйпи - в об'єкт кластера до тих пір, поки не досягається максимальна місткість об'єкта, потім створюється інший об'єкт для додаткових кусків даних.

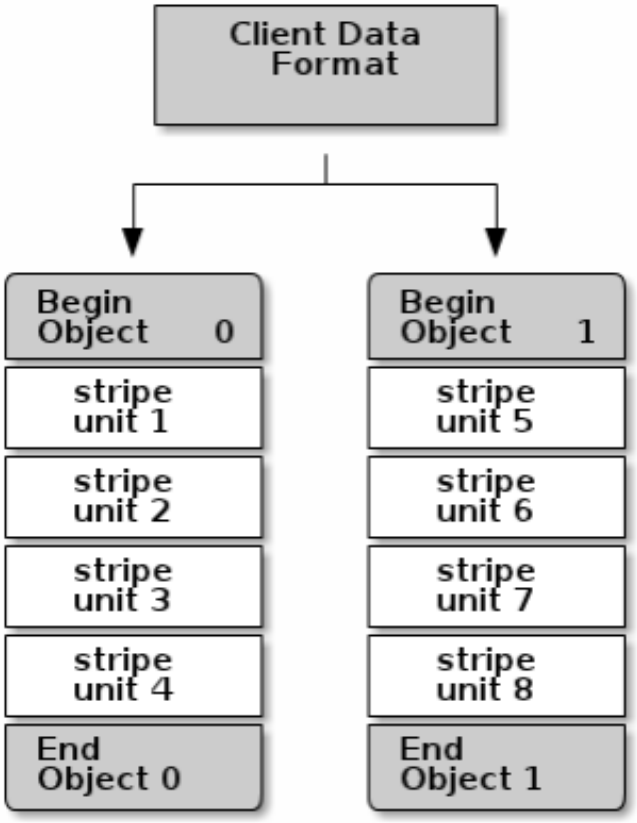

Рисунок 2.13 - Чергування

Найпростіша форма чергування може бути ефективна для маленьких блокових пристроїв, S3 або об'єктів Swift і файлів CephFS. Проте, ця проста форма не має всіх переваг здатності Ceph розподіляти дані по групах розміщення, отже, не дуже сильно покращує продуктивність.

Якщо передбачається використання файлів великих розмірів, великих S3 або Swift об'єктів (наприклад, відео), або великих директорій CephFS, можна помітити значне поліпшення продуктивності операцій читання / запису при чергуванні даних клієнта по різних об'єктах. Значне поліпшення продуктивності при записі відбувається, коли клієнт записує страйпи даних, які чергуються, до відповідних об'єктів паралельно. Після того, як об'єкти присвоюються різним групам розміщення і різним OSD, операції запису відбуваються паралельно на максимальній швидкості. Чергування відбувається незалежно від реплікації. Як тільки CRUSH починає реплікувати об'єкт за всіма OSD, страйпи реплікуються автоматично.

Таким чином, кластер зберігання даних Ceph динамічний, як живий організм. У той час як у багатьох пристроях зберігання не повністю використовуються ресурси процесора і оперативної пам'яті стандартного сервера, Ceph використовує все.

2.6 Висновки до другого розділу

У другому розділі описано архітектуру Ceph (об’єктне сховище, блоковий пристрій, ФС) та процес зберігання даних. Також проаналізовано масштабованість і високу доступність моніторів та автентифікації. Докладно описано основні складові динамічного управління кластером (пули даних, присвоєння груп розміщення і призначення їх ID, ребалансування). Проаналізовано процес чергування даних.

3 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Встановлення платформи OpenStack

Для установки хмарної платформи OpenStack була використана утиліта під назвою packstack [24]. Вона дозволяє швидко розгорнути OpenStack як в інтерактивному, так і в НЕ інтерактивному форматі, використовуючи заздалегідь заготовлений файл з відповідями. Була використана установка в НЕ інтерактивному форматі, так як в такому випадку налаштування установки задокументовані. В процесі інсталяції були виконані кроки, наведені нижче по тексту розділу.

1. Встановлення ОС Fedora 20.
2. Підключення репозиторіїв OpenStack:

```
yum install http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm
```

3. Установка пакета openstack-packstack, що містить утиліту packstack:

```
yum install -y openstack-packstack
```

4. Генерація SSH ключів для полегшення доступу до вузлів Nova Compute з вузла Controller: ssh-keygen
5. Перегляд опцій команди packstack:

```
packstack -h | less
```

6. Генерація файлу з відповідями з налаштуваннями за замовчуванням:

```
packstack --gen-answer-file /root/answers.txt
```

7. Зміна файлу /root/answers.txt.
8. Установка OpenStack:

```
packstack --answer-file /root/answers.txt
```

Так як за замовчуванням OpenStack не має можливості забезпечення високої доступності (High-Availability), автором був розроблений відповідний скрипт на bash. Скрипт дозволяє при падінні гіпервізора перезапускати

віртуальні сервери на тих вузлах OpenStack ,що залишилися в роботі за умови наявності системи зберігання даних спільного доступу.

Його лістинг наведено в Додатку Б: Лістинг скрипта HA для OpenStack.

Отримати доступ до порталу самообслуговування можна, набравши адресу вузла контролера в веб-браузері [25]. При логін в систему виводиться статистика використання, яку можна побачити на рисунку 3.1.

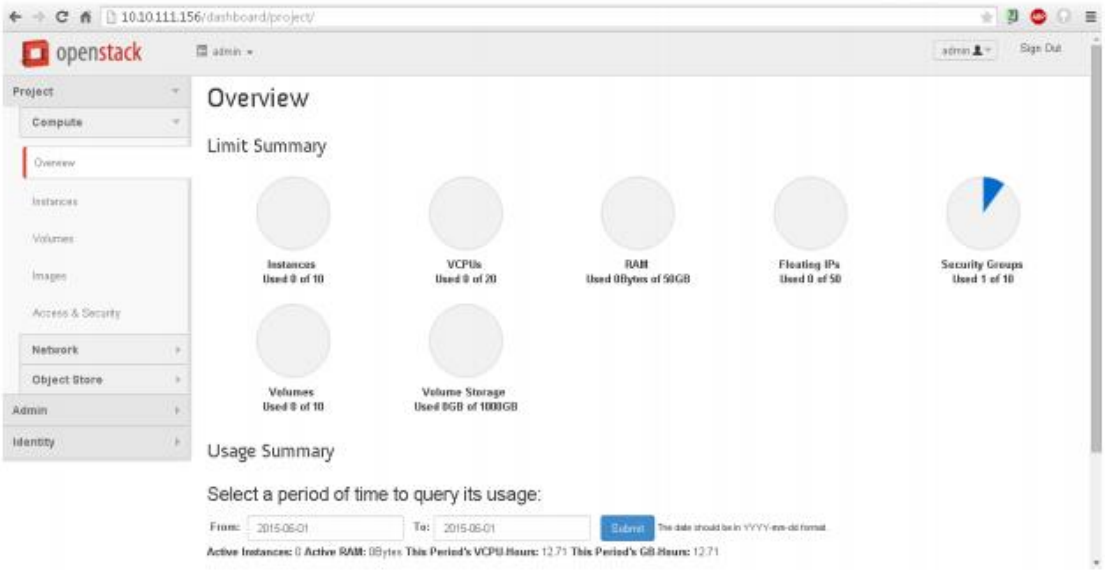

Рисунок 3.1 – Статистика використання

3.2 Розгортання кластера Ceph

Всі кластери Ceph вимагають наявність як мінімум одного монітора і стільки OSD, скільки копій об'єкта повинно зберігатися в кластері. Установка моніторів є першим кроком у розгортанні кластера зберігання даних Ceph. Розгортання моніторів також встановлює важливі критерії для всього кластера, такі як кількість реплік для пулів, кількість груп розміщення на кожному OSD, інтервали між перевірками аутентифікації, і т.д. [25]

Розгортання монітора вимагає ряд попередніх налаштувань:

– унікальний ідентифікатор: fsid - це унікальний ідентифікатор кластера, File System ID;

– ім'я кластера: Кластери Ceph мають імена, які представляють собою просту рядок без пробілів. За замовчуванням ім'я кластера ceph, але його можна змінити. Перезапис імені кластера за замовчуванням особливо корисний, коли робота ведеться з багатьма кластерами, і потрібно чітке розуміння, з яким саме кластером ведеться робота. Для ідентифікації імені кластера через інтерфейс командного рядка потрібно змінити назву конфігураційного файлу Ceph за допомогою імені кластера (наприклад, ceph.conf, us-west.conf, us-east.conf, і т.д.);

– ім'я монітора: Кожен екземпляр монітора в кластері має унікальне ім'я. Як правило, ім'я монітора Ceph це ім'я хоста. Коротку версію імені хоста можна отримати за допомогою команди hostname –s;

– карта моніторів: Розгортання моніторів вимагає генерацію карти моніторів. Карта моніторів включає в себе fsid, ім'я кластера і ім'я і IP адресу принаймні одного хоста;

– Keyring монітора: Монітори взаємодіють один з одним за допомогою секретного ключа. Необхідно згенерувати keyring з ключем монітора і надати його при установці моніторів;

– Keyring адміністратора: Для використання ceph утиліт командного рядка необхідна наявність client.admin користувача. Необхідно згенерувати адмін користувача і keyring, потім додати client.admin користувача в keyring монітора;

В ході установки кластера Ceph автором були виконані наступні кроки:

1. Інсталяція необхідних пакетів ceph на всіх трьох хостах моніторів (test1, test3, test5).

```
yum·install·ceph
```

Наступні дії виконуються на хості test1.

2. Перевірка наявності директорії для конфігураційного файлу Ceph.

За замовчуванням, Ceph використовує / etc / ceph. . Під час установки ceph директорія / Etc / ceph створюється автоматично.

ls / etc / ceph

3. Створення конфігураційного файлу Ceph. За замовчуванням Ceph використовує ceph.conf, де ceph відображає ім'я кластера.

touch /etc/ceph/ceph.conf

4. Генерація унікального ID (fsid) для кластера.

uuidgen

5. Додавання унікального ID в файл конфігурації Ceph.

vi /etc/ceph/ceph.conf

fsid = 4b86c306-b68d-48e7-8007-c78ef354754c

6. Додавання списку моніторів в файл конфігурації Ceph.

mon initial members = test1, test3, test5

7. Додавання IP адреси всіх моніторів в файл конфігурації і збереження файлу.

mon host = 10.10.111.93,10.10.111.134,10.10.111.141

Зауваження: Можливо використовувати IPv6 адреси теж, але опція ms bind ipv6 повинна бути в значенні true.

8. Створення keyring для кластера і генерація секретного ключа монітора.

ceph-authtool --create-keyring /tmp/ceph.mon.keyring --gen-key -n mon. -

cap mon 'allow *'

9. Генерація keyring адміністратора, генерація користувача client.admin і додавання його в keyring.

ceph-authtool --create-keyring /etc/ceph/ceph.client.admin.keyring --gen-key -n client.admin --set-uid = 0 --cap mon 'allow *' --cap osd 'allow *' --cap mds 'Allow'

10. Додавання ключа client.admin в ceph.mon.keyring.

ceph-authtool

/tmp/ceph.mon.keyring

--import-keyring

/etc/ceph/ceph.client.admin.keyring

11. Генерація карти монітора з використанням імені та IP адреси хоста і FSID. Збереження її як

monmaptool --create --add test1 10.10.111.93 --add test3 10.10.111.134 - add test5 10.10.111.141 --fsid 4b86c306-b68d-48e7-8007-c78ef354754c / Tmp / monmap

12. Створення директорії за замовчуванням для даних на хостах моніторів (test1, test3, test5 відповідно).

mkdir / var / lib / ceph / mon / ceph-test1

mkdir / var / lib / ceph / mon / ceph-test3

mkdir / var / lib / ceph / mon / ceph-test5

13. Копіювання файлів аутентифікації на два, що залишилися хоста моніторів.

scp / tmp / monmap /tmp/ceph.mon.keyring test3: / tmp /

scp / tmp / monmap /tmp/ceph.mon.keyring test5: / tmp /

14. Відправка демона монітора карти монітора і keyring (на кожному хості монітора відповідно).

ceph-mon --mkfs -i test1 --monmap / tmp / monmap --keyring /tmp/ceph.mon.keyring

ceph-mon --mkfs -i test3 --monmap / tmp / monmap --keyring /tmp/ceph.mon.keyring

ceph-mon --mkfs -i test5 --monmap / tmp / monmap --keyring /tmp/ceph.mon.keyring

15. Код остаточного формування файлу конфігурації Ceph наведено в лістингу 3.1.

16. Копіювання файлу конфігурації на всі хости моніторів.

scp /etc/ceph/ceph.conf test3: /etc/ceph/ceph.conf

scp /etc/ceph/ceph.conf test5: /etc/ceph/ceph.conf

17. Створення файлу done, що відображає готовність монітора до запуску (На кожному хості моніторів відповідно).

sudo touch / var / lib / ceph / mon / ceph-test1 / done

sudo touch / var / lib / ceph / mon / ceph-test3 / done

sudo touch / var / lib / ceph / mon / ceph-test5 / done

Лістинг 3.1 - Формування файлу конфігурації Ceph

```
[Global]
fsid = 4b86c306-b68d-48e7-8007-c78ef354754c
mon initial members = test1, test3, test5
mon host = 10.10.111.93,10.10.111.134,10.10.111.141
public network = 10.10.111.0/24
cluster network = 10.10.111.0/24
auth cluster required = cephx
auth service required = cephx
auth client required = cephx
osd journal size = 1024
filestore xattr use omap = true
osd pool default size = 3
osd pool default min size = 2
osd pool default pg num = 333
osd pool default pgp num = 333
osd crush chooseleaf type = 1
[Mon.test1]
host = test1
[Mon.test3]
host = test3
[Mon.test5]
host = test5
```

18. Запуск моніторів (на кожному хості відповідно).

/etc/init.d/ceph start mon.test1

/etc/init.d/ceph start mon.test3

/etc/init.d/ceph start mon.test5

19. Перевірка, чи створив Ceph дефолтні пули.

ceph osd lspools

Маємо такий висновок:

0 data, 1 metadata, 2 rbd,

20. Перевірка роботи кластера.

ceph -s

Було отримано наступний висновок, що говорить про те, що монітор запущений і містить кілька помилок здоров'я, які говорять про те, що групи розміщення неактивні, код наведено в лістингу 3.2.

Лістинг 3.2 – Висновок за результатами інсталяції Ceph

```
cluster 4b86c306-b68d-48e7-8007-c78ef354754c
health HEALTH_ERR 192 pgs stuck inactive; 192 pgs stuck
unclean;
no osds monmap
e1: 3 mons at
{Test1 = 10.10.111.93: 6789/0, test3 = 10.10.111.134: 6789/0,
test5 = 10.10.111.141},
election epoch 4, quorum 0,1,2 test1, test3, test5
osdmap e1: 0 osds: 0 up, 0 in
pgmap v2: 192 pgs, 3 pools, 0 bytes data, 0 objects
0 kB used, 0 kB / 0 kB avail
192 creating
```

Як тільки OSD будуть додані і запущені, помилки груп розміщення зникнуть.

Подальші кроки були виконані на кожному хості кластера test1, test2, test3, test4, test5.

1. Установка необхідних пакетів ceph на хости test2 і test4 yum install ceph

2. Створення OSD. Можна вручну вказати UUID, в іншому випадку він буде встановлений автоматично (відлік починається з 0). Наступна команда виведе номер OSD, який буде використовуватися у всіх подальших кроках

ceph osd create

Відповідно, на test1 будуть osd.0 і osd.1, на test2 - osd.2, osd.3, на test3 - osd.4, osd.5, на test4 - osd.6, osd.7, на test5 - osd.8, osd.9.

3. Створення директорії на новому OSD.

4. Підготовка пристрою до використання Ceph шляхом створення на ньому ФС і монтування в створену вище директорію. Дана операція виконується для кожного пристрою в кластері за аналогією.

mkfs -t xfs / dev / vdc

mount / dev / vdc / var / lib / ceph / osd / ceph-0

5. Ініціалізація директорії даних OSD (для всіх OSD в кластері по аналогії). Директорія повинна бути порожньою перед запуском ceph-osd з

опцією --mkkey.

6. Реєстрація ключа аутентифікації OSD (для всіх OSD в кластері за аналогією).

7. Додавання нод Ceph в CRUSH карту (для всіх хостів в кластері за аналогією).

8. Перенесення нод Ceph в корінь ієрархії (для всіх хостів в кластері за аналогією).

9. Додавання OSD в CRUSH карту, після чого він зможе отримувати дані

10. Запуск OSD.

11. Створення порожніх файлів для автозапуску демонів OSD при старті системи.

Кластер не увійде в стан active + clean, поки в ньому не буде досить OSD для підтримки необхідної кількості копій об'єкта. Після розгортання у кластера є дефолтна CRUSH карта, тим не менш, вона не містить Ceph OSD демонів [23].

Так як у вимогах в до системи вказана можливість використання твердотільних накопичувачів для більш вимогливих до продуктивності системи зберігання клієнтів, було прийнято рішення налаштування cache-tier на кластері.

Кеш тир забезпечує більш швидкий доступ до даних звичайних пулів. Це забезпечується за рахунок створення відносно невеликого за розміром пулу з швидких дисків (SSD) і налаштування його як кеша для основного пулу з повільних обертових дисків. Це дозволяє значно прискорити операції читання-запису, зберігаючи при цьому високу дискову ємність пулу. При цьому клієнт фактично працює з основним пулом (так званим холодним), а кеш пул виконує прозоре проксування клієнтських запитів до основного пулу. [27]

Необхідно прискорити пул з повільних дисків cold_storage за рахунок пулу з швидких SSD дисків hot_storage. Попередньо потрібно написати кастомну (призначену для користувача) CRUSH-карту, щоб пул hot_storage складався тільки з SSD-дисків, а пул cold_storage -тільки з повільних дисків. У

нашому кластері osd.0 на test1, osd.2 на test2, osd.4 на test3, osd.6 на test4 є SSD дисками.

Тепер CRUSH-правило номер 0 відноситься тільки до повільних дисків, а правило номер 1 тільки до SSD.

Завантаження нової карти в кластер відбувається за командою

ceph osd setcrushmap - i crushmap.bin.

Створення пулів для подальшої інтеграції з OpenStack. За замовчуванням блокові пристрої Ceph використовують пул rbd. Можна використовувати будь-який доступний пул, але рекомендується створювати окремі пули для Cinder і для Glance.

Налаштування cache-tier. Після цієї команди пул hot_storage почне працювати як кеш пулу vms:

ceph osd tier add vms hot_storage.

Далі потрібно вказати режим, в якому буде працювати кеш. Існує два режиму роботи кеша.

1. Writeback. Він є основним режимом роботи кеша. Коли клієнт надсилає запит на запис до основного пулу, кеш перехоплює його і записує до себе. Потім кеш поступово витісняє «брудні» дані в основний пул, виконуючи при цьому послідовну запис, що значно прискорює процес витіснення (flushing).

Коли клієнт надсилає запит на читання, а потрібних даних немає в кеші, він підвантажує їх з холодного сховища. Це дає суттєвий приріст продуктивності на читання для часто використовуваних блоків даних, однак навпаки знижує швидкість читання для блоків даних, які потрібні рідко.

2. Read-only . В цьому режимі клієнт пише відразу на холодне сховище, а читає з кеш пулу. Дані по необхідності завантажуються в кеш. Даний режим в основному підходить для даних, які рідко або ніколи не змінюються, наприклад, фото і відео в соціальних мережах і на загальних ресурсах.

Очевидно, що даний режим не дає ніякого приросту на запис, проте потенційно прискорює операції читання.

В даному випадку було обрано режим writeback:

ceph osd tier cache-mode hot-storage writeback

ceph osd tier set-overlay vms hot-storage

Налаштування кеш tier завершена.

3.3 Інтеграція хмарної платформи OpenStack з кластером Ceph

Можна використовувати образи Ceph Block Device з OpenStack через libvirt, який налаштовує інтерфейс QEMU для використання librbd (рисунок 3.2). [28].

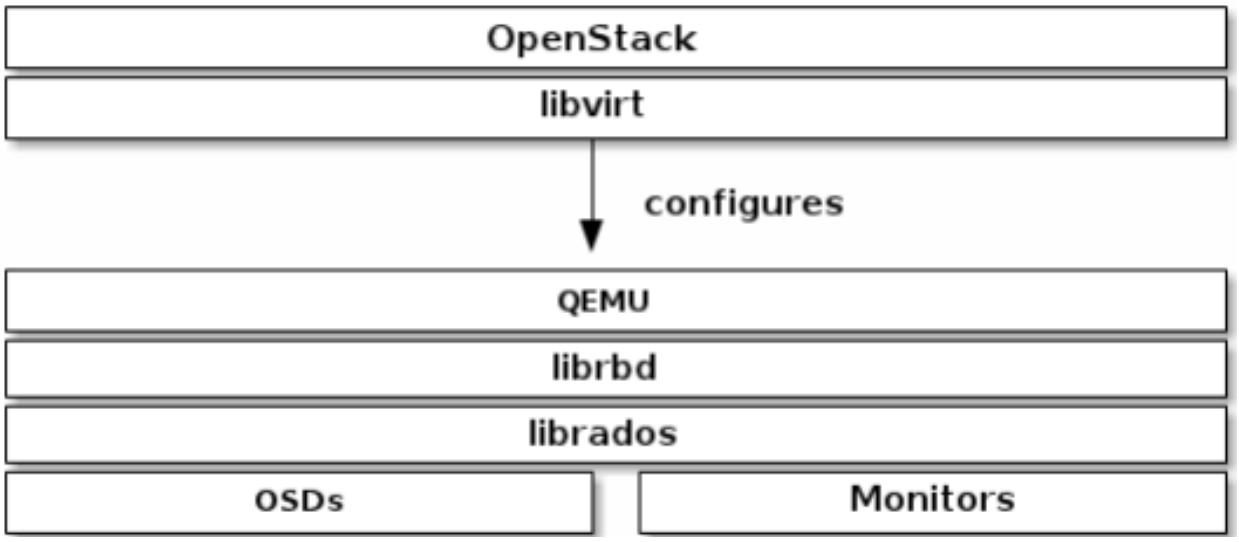

Рисунок 3.2 – Використання образів Ceph Block Device з OpenStack

Три частини OpenStack інтегруються з блочними пристроями Ceph [29,30]:

– образи: OpenStack Glance управляє образами операційних систем для віртуальних машин. Образи є незмінними;

– диски: Диски є блоковими пристроями. OpenStack використовує диски для завантаження віртуальних машин або приєднує диски наявних віртуальнх машин. OpenStack управляє дисками за допомогою сервісу Cinder;

– ефемерні диски: є тимчасовими дисками операційної системи. За замовчуванням, ці диски з'являються як файл в ФС гіпервізора (зазвичай в директорії / var / lib / nova / instances / <uuid> /) при завантаженні віртуальної машини.

До OpenStack Havana, єдиним способом завантажити віртуальну машину в Ceph було використання функціональності Cinder завантаження-з-диска. Тим не менше, на даний момент є можливим завантажувати кожну віртуальну машину всередині Ceph безпосередньо без використання Cinder, що дозволяє легко виробляти управління операціями за допомогою процесу динамічної міграції. Як доповнення, якщо гіпервізор перестає відповідати, дана функція також зручна для запуску nova evacuate і безпроблемного запуску віртуальної машини в іншому місці.

Рисунок 3.3 відображає, як компоненти OpenStack взаємодіють з Ceph.

Можна використовувати OpenStack Glance для зберігання образів в Ceph Block Device, і використовувати Cinder для завантаження віртуальних машин з використанням copy-on-write клону образу [31, 32].

Ceph не підтримує QCOW2 для зберігання в якості диска віртуальної машини. Тому для запуску віртуальної машини в Ceph формат образу Glance повинен бути RAW.

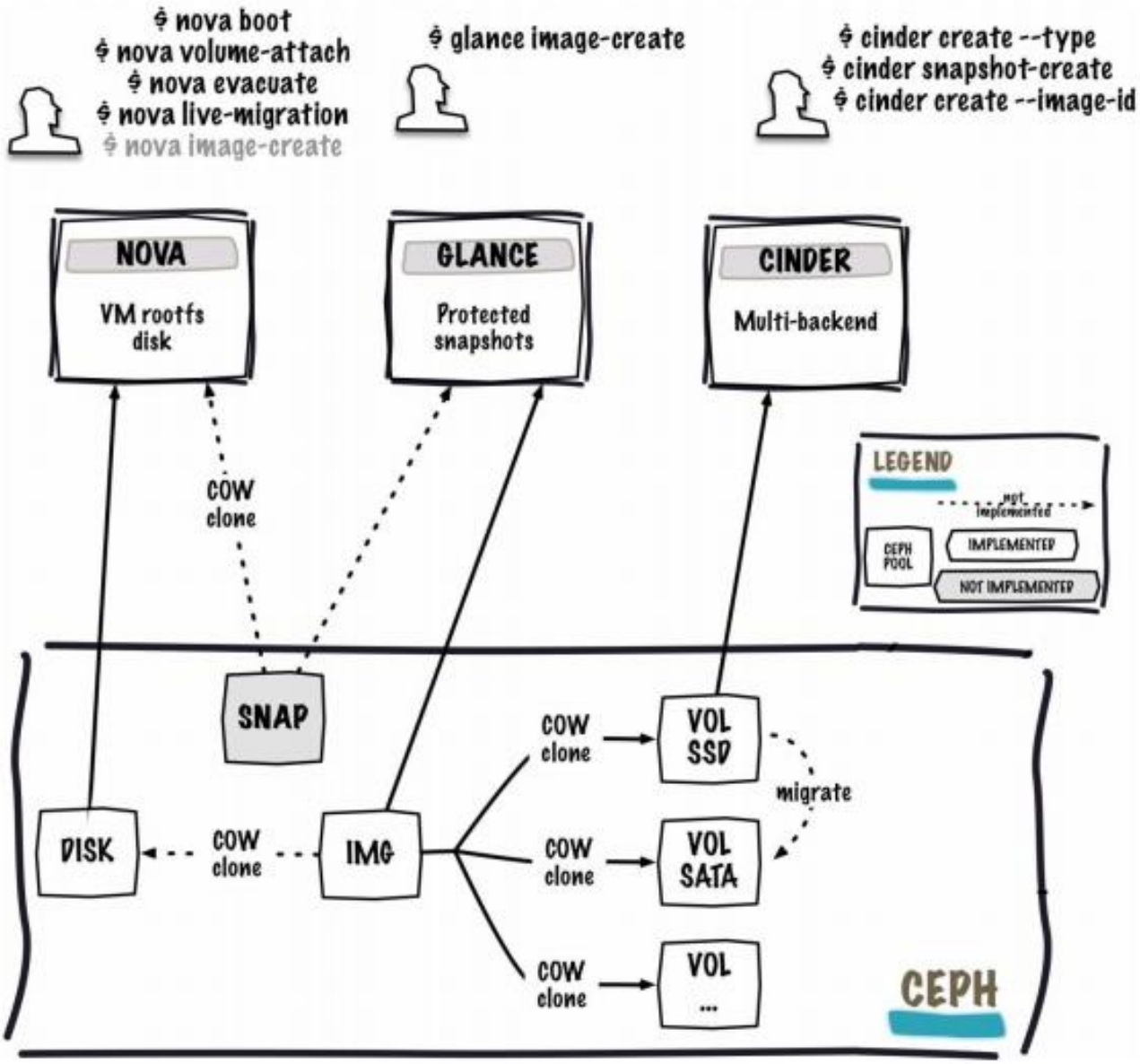

Рисунок 3.3 – Взаємодія компонентів OpenStack з Ceph

Необхідно навести ті кроки, які були виконані для налаштування Glance, Cinder і Nova.

3.3.1 Налаштування OpenStack клієнтів Ceph

Вузли, на яких запущені glance-api, cinder-volume, nova-compute і cinder-backup виступають в якості клієнтів Ceph. Кожен з них вимагає наявності ceph.conf файлу. Програмний код, який відповідає за копіювання з вузла test1 файлу конфігурації на вузли OpenStack:

```
ssh 10.10.111.156 sudo tee /etc/ceph/ceph.conf </etc/ceph/ceph.conf
ssh 10.10.111.170 sudo tee /etc/ceph/ceph.conf </etc/ceph/ceph.conf
```

Установка пакетів на хости клієнтів Ceph. Вузол з IP-адресою 10.10.111.156 є контролером в OpenStack, на ньому запущені сервіси glance-api і cinder-backup. Вузол з IP-адресою 10.10.111.170 є обчислювальним вузлом в OpenStack, на ньому запущені сервіси nova-compute і cinder-volume.

На вузлі з glance-api необхідні пакети Python для librbd:

```
sudo yum install python-rbd
```

На вузлах з nova-compute, cinder-backup і cinder-volume необхідні пакети Python і утиліти командного рядка:

```
sudo yum install ceph
```

Для налаштування аутентифікації клієнтів Ceph докладніші відомості виконані на хості test1. Так як в створеному вище кластері Ceph включена аутентифікація cephx, необхідне створення нових користувачів для Nova / Cinder і Glance.

Також потрібно секретний ключ користувача client.cinder з libvirt. Процес libvirt вимагає його для доступу до кластеру в процесі приєднання блочного пристрою з Cinder.

Створення тимчасової копії секретного ключа на вузлі з nova-compute:

```
ceph auth get-key client.cinder | ssh 10.10.111.170 tee client.cinder.key.
```

UUID ключа був збережений для подальшого налаштування nova-compute. Використання одного і того ж UUID потрібно з точки зору узгодженості платформи.

3.3.2 Налаштування OpenStack для використання Ceph

Налаштування Glance. Glance може використовувати множинні бекенди для зберігання образів. Для використання блокових пристроїв Ceph за замовчуванням, необхідні наведені нижче налаштування Glance. [33]

Код редагування /etc/glance/glance-api.conf і додавання секції [Glance_store] наведено в лістингу 3.3.

Лістинг 3.3 – Код редагування і додавання секції

```
[DEFAULT]
...
default_store = rbd
...
[Glance_store]
stores = rbd
rbd_store_pool = images
rbd_store_user = glance
rbd_store_ceph_conf = /etc/ceph/ceph.conf
rbd_store_chunk_size = 8
```

Додавання наступного рядка в секцію [DEFAULT] для використання клонування образів в режимі copy-on-write:

show_image_direct_url = True

Відключення Glance кеш менеджменту для уникнення кешування образів / var / lib / glance / image-cache /, видалення з конфігураційного файлу рядки flavor = keystone + cachemanagement і додавання наступного:

[Paste_deploy]

flavor = keystone

Налаштування Cinder. OpenStack вимагає драйвер для взаємодії з блочними пристроями Ceph. Також необхідно вказати ім'я пулу для блочного пристрою.

Редагування /etc/cinder/cinder.conf на вузлах OpenStack відбувається за допомогою коду, наведеного в лістингу 3.4.

Так як в кластері використовується аутентифікація cephx, необхідно налаштувати користувача і UUID ключа, який був доданий в libvirt раніше:

rbd_user = cinder

rbd_secret_uuid = 457eb676-33da-42ec-9a8c-9293d545c337

Лістинг 3.4 – Код редагування на вузлах

```
volume_driver = cinder.volume.drivers.rbd.RBDDriver
rbd_pool = volumes
rbd_ceph_conf = /etc/ceph/ceph.conf
rbd_flatten_volume_from_snapshot = false
rbd_max_clone_depth = 5
rbd_store_chunk_size = 4
rados_connect_timeout = -1
glance_api_version = 2
```

Налаштування Cinder Backup. На вузлі Cinder Backup редагування /etc/cinder/cinder.conf шляхом додавання коду, наведеного в лістингу 3.5:

Лістинг 3.5 – Код налаштування Cinder Backup

```
backup_driver = cinder.backup.drivers.ceph
backup_ceph_conf = /etc/ceph/ceph.conf
backup_ceph_user = cinder-backup
backup_ceph_chunk_size = 134217728
backup_ceph_pool = backups
backup_ceph_stripe_unit = 0
backup_ceph_stripe_count = 0
restore_discard_excess_bytes = true
```

Налаштування Nova. Для використання пристроїв Cinder (як звичайного блокового пристрою, так і при завантаженні з диска) необхідно вказати в налаштуваннях Nova, який користувач і UUID використовуються при приєднанні пристрою. Libvirt буде посилатися на цього користувача при підключенні до кластеру Ceph.

rbd_user = cinder

rbd_secret_uuid = 457eb676-33da-42ec-9a8c-9293d545c337

Ці два параметри також використовуються ефемерними дисками Nova. Для завантаження всіх віртуальних машин безпосередньо в Ceph, необхідно налаштувати ефемерний бекенд для Nova.

Передусім потрібно увімкнути використання сокета адміністрування, так як це допомагає в дослідженні проблем.

Доступ до сокету можна отримати наступним чином:

```
ceph daemon /var/run/ceph/ceph-client.cinder.19195.32310016.asok help
```

На compute вузлі редагування конфігураційного файлу Ceph:

```
[Client]
rbd cache = true
rbd cache writethrough until flush = true
admin socket = /var/run/ceph/$cluster-$type.$id.$pid.$cctid.asok
```

На compute вузлі редагування /etc/nova/nova.conf в секції [libvirt] шляхом додавання:

```
[Libvirt]
images_type = rbd
images_rbd_pool = vms
images_rbd_ceph_conf = /etc/ceph/ceph.conf
rbd_user = cinder
rbd_secret_uuid = 457eb676-33da-42ec-9a8c-9293d545c337
```

Також вважається хорошою практикою відключати ін'єкцію. Під час завантаження інстанси Nova зазвичай намагається відкрити кореневу ФС на віртуальній машині. Потім Nova вносить значення, такі як пароль, ssh ключі і т.д. безпосередньо в ФС. Проте, краще покладатися на сервіс метаданих і cloud-init. [33]

На compute вузлі редагування /etc/nova/nova.conf шляхом додавання наступних рядків в секцію [libvirt]:

```
inject_password = false
inject_key = false
inject_partition = -2
```

Додавання вказаних нижче прапорців для забезпечення правильної живої міграції:

```
live_migration_flag = "VIR_MIGRATE_UNDEFINE_SOURCE,
VIR_MIGRATE_PEER2PEER, VIR_MIGRATE_LIVE,
VIR_MIGRATE_PERSIST_DEST "
```

Перезапуск OpenStack. Для активації драйвера блочного пристрою Ceph і завантаження пулу блочного пристрою в конфігурацію необхідно перезапустити OpenStack.

Наступні команди були виконані автором на відповідних вузлах:

```
sudo service openstack-glance-api restart
sudo service openstack-nova-compute restart
sudo service openstack-cinder-volume restart
sudo service openstack-cinder-backup restart
```

Як тільки OpenStack увімкнеться знову, створення диска і завантаження з нього стає можливим.

Завантаження з блочного пристрою. Створення диска, використовуючи утиліту командного рядка Cinder:

```
cinder create --display-name test_volume 100
```

Диски можна створювати з образів. Образ повинен мати RAW формат.

Можна використовувати qemu-img для конвертації з одного формату в інший. Наприклад:

```
qemu-img convert -f {source-format} -O {output-format} {source-
filename} {output-filename}
qemu-img convert -f qcow2 -O raw precise-cloudimg.img precise-
cloudimg.raw
```

Коли Glance і Cinder використовують блокові пристрої Ceph, образ є copy-on-write клоном, тому з нього можна швидко створити новий диск. У порталі самообслуговування OpenStack можна завантажитися з цього диска, виконавши наступні дії: [35]

– завантаження нової інстанси;

– вибір способу, пов'язаного з copy-on-write клоном;

– вибір опції «завантаження з диска»;

– вибір створеного диску.

3.4 Висновки до третього розділу

У третьому розділі описано процес інсталяції платформи OpenStack. Покроково наведено всі етапи розгортання кластера Ceph. Відображено особливості інтеграції хмарної платформи OpenStack з кластером Ceph, зокрема налаштування OpenStack для клієнтів та використання Ceph.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Небезпечні й шкідливі фактори при виконанні робіт за комп'ютером

За даними вчених [37], у користувачів ПК за кілька годин до закінчення робочого часу передчасно знижується працездатність і розвивається втома. У 48% обстежених працівників обчислювальних центрів було виявлено невротичні розлади: погіршення пам'яті, знесилення, серцебиття, негативні дисфункції, неуважність, неврастенія.

В таблиці 4.1 вказані потенційно небезпечні на організм людини фактори при виконанні робіт за комп'ютером

Таблиця 4.1 - Потенційно небезпечні виробничі фактори

Небезпечний фактор	Фактичні дані замірів	Нормовані значення
Рентгенівське випромінювання	12-15 мкР/год	75 мкР/год
Ультрафіолетове випромінювання	0-0.005 Вт/м^2	0.01 Вт/м^2 0.01 Вт/м^2
Видимий діапазон	0.1-2.0 Вт/м^2 2.5-4.0 Вт/м^2	10.0 Вт/м^2
Яскравість	75-80 кД/м^2	>50кД/м^2
ІЧ-випромінювання	0.15-5.0 Вт/м^2	100.0 Вт/м^2
Електростатичне поле	15 Вт/м	20-60 кВ/м
Шум	45-55 дБА	60 дБА

Важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення, які визначені у НПАОП 0.00-7.15-18. Організація робочого місця повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Відстань від екрана до ока фахівців, які працюють за комп'ютером, визначається вимогами ДСанПіН 3.3.2.007-98.

Оскільки екран монітора є джерелом світла, тому необхідно організувати освітлення та розташувати його так, щоб у полі зору користувача не було інших яскравих джерел світла, а також освітленість екрану не збільшувалась за рахунок будь-якого стороннього джерела світла. Пофарбовані у світлі кольори меблі приміщення і великі вікна є додатковими джерелами світла. У надто освітленому приміщенні погано видно невеликі букви і цифри на екрані монітора. Це викликає головний біль, погіршення зору, зниження концентрації, а також призводить до помилок у роботі через некоректне сприйняття інформації. Спектр випромінювання комп'ютера містить у собі рентгенівську, ультрафіолетову й інфрачервону області спектра, а також широкий діапазон електромагнітних хвиль інших частот. Небезпека рентгенівських променів вважається зараз спеціалістами досить малою, оскільки цей вид променів поглинається склом екрана. Проте необхідно пам'ятати, що рентгенівське випромінювання, навіть мізерно малих інтенсивностей, сприяє іонізації повітря. Інфрачервоне випромінювання шкідливо діє на зір, втомлюючи очі, при тривалій дії порушує нормальне сприйняття кольору оком. Рентгенівське випромінювання шкідливо впливає на кісткові тканини і кровотворні функції кісткового мозку. Електромагнітне випромінювання, яким супроводжується показ зображення на моніторі комп’ютера, негативно впливає на органи зорового сприйняття людини. Основні джерела електромагнітного випромінювання вказані в таблиці 4.2 [38].

Таблиця 4.2 - Основні джерела електромагнітного випромінювання

Джерело	Діапазон частот (перша гармоніка)
Монітор мережний трансформатор блока живлення	50 Гц
Статичний перетворювач напруги в імпульсному блоці живлення	20 - 100 кГц
Анодна напруга монітора, (тільки для моніторів із ЭЛТ)	0 Гц (електростатика)
Системний блок (процесор)	50 Гц - 1000 МГц
Джерела безперебійного живлення	50 Гц, 20 - 100 кГц

Необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Відеотермінали випромінюють електромагнітні хвилі в дуже широкому діапазоні. У радіодіапазоні вони продукуються катодною трубкою; основне же джерело - горизонтальні і вертикальні відхиляючі котушки, які забезпечують сканування електронного променя по екрані в діапазоні 15 - 35 кГц. На відстані 50 мм від екрана напруженість електричного поля має значення від одиниць до 10 В/м, а магнітна індукція - від 10-8 до 10-7 Тл. Відеотермінали випромінюють також змінні електричні і магнітні поля з частотою 50 або 100 Гц і їхні гармоніки. [39]

В таблиці 4.3 наведено гігієнічні норми впливу високовольтного електромагнітного поля (ЕМП).

Таблиця 4.3 - Гігієнічні норми впливу високовольтного ЕМП

Напруженість електричного поля, В/м	Час перебування людини в електричному полі протягом однієї доби, хв.
менше 5	Без обмеження
від 5 до 10	Не більш 180
від 10 до 15	Не більш 90
від 15 до 20	Не більш 10
від 20 до 25	Не більш 5

Під час роботи комп'ютерів суттєво зменшується в робочій зоні кількість негативно заряджених іонів. Це обумовлено дією електричних полів, насамперед електростатичного. За санітарними нормами рівень концентрації аероіонів, відхилення від якого складає небезпеку здоров'ю людини знаходиться в діапазоні від 600 АІ (аероіонів) до 50000 АІ в куб.см.

В таблиці 4.4 наведено рівні іонізації повітря приміщень при роботі на ПК.

Таблиця 4.4 - Рівні іонізації повітря приміщень при роботі на ПК

Рівні	Кількість іонів в 1 $см^2$ куб. повітря	
	n^+	n^-
Мінімально необхідні	400	600
Оптимальні	1500 - 3000	3000 - 5000
Максимально допустимі	50000	50000

Тривале вдихання повітря, що містить надлишок позитивних іонів, приводить до головної болі і запаморочення, до так званого "синдрому постійної втоми. Збільшується pH крові, порушується процес протікання хімічних реакцій на клітинному рівні (метаболізм). Надлишок позитивних іонів - причина функціональних порушень роботи щитовидної залози, порушення обміну речовин. На цьому фоні виникають депресії, стани тривоги і занепокоєння, безсоння.

4.2 Ергономічні вимоги до робочого місця користувача ПК

Робоче місце — це зона простору, що оснащена необхідним устаткуванням, де відбувається трудова діяльність одного працівника чи групи працівників [38].

Раціональне планування робочого місця має забезпечувати: найкраще розміщення знарядь і предметів праці, не допускати загального дискомфорту, зменшувати втомлюваність працівника, підвищувати його продуктивність праці. Площа робочого місця має бути такою, щоб працівник не робив зайвих рухів і не відчував незручності під час виконання роботи. Важливо мати також можливість змінити робочу позу, тобто положення корпуса, рук, ніг. Проте доцільно виключати або мінімізувати всі фізіологічно неприродні і незручні

положення тіла. Проведені дослідження показують, що при раціональній організації робочих місць продуктивність праці зростає на 15–25%. [39]

Організація робочого місця користувача ПК має відповідати ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги», ДСан ПіН 3.3.2.007-98, характеру та особливостям трудової діяльності.

Площа одного робочого місця користувача ПК повинна складати не менше 6 м2, а об'єм – не менше 20 м3. Конструкція робочого місця користувача ПК повинна відповідати сучасним вимогам ергономіки, характеру виконуваної роботи і забезпечити оптимальне розміщення на робочій поверхні документів та обладнання ПК (монітора, системного блоку, клавіатури, мишки та інших периферійних пристроїв. Монітор на робочому місці встановлюється так, щоб верхній край екрана знаходився на рівні очей.

Розташування монітора ПК має забезпечувати:

– безпечність роботи в цілому;

– зручність та ефективність зорової роботи з екраном в вертикальній площині під кутом 300 від лінії зору, площина екрана при цьому має бути перпендикулярною нормальній лінії зору користувача.

Клавіатура розміщується на поверхні столу або висувній полиці на відстані 100-300мм від краю, ближчого до користувача. Кут нахилу клавіатури має бути в межах 5-150. Поверхня клавіатури повинна бути матовою з коефіцієнтом відбиття 0,4. клавіші клавіатури мають бути зручними в роботі і м'якими при натисканні (хід всіх клавіш має бути одинаковим з мінімальним опором натискання 0,25Н та максимальним – не більше 1,5Н) [40].

При розміщенні робочих місць з ПК слід дотримуватися вимог, зазначених в ДНАОП 0.00-1.31-99:

– робочі місця розміщуються на відстані не менше 1м від стін з світловими прорізями; – відстань між бічними поверхнями моніторів ПК має

бути не менше 1,2м; – відстань між тильною поверхнею монітора одного ПК та екраном монітора іншого ПК має бути не меншою 2,5м.

Вимоги двох останніх пунктів враховуються також при розміщенні робочих місць з ПК в суміжних приміщеннях з урахуванням конструктивних особливостей стін та перегородок.

Загальні принципи організації робочого місця:

– на робочому місці не повинно бути нічого зайвого. Усі необхідні для роботи предмети мають бути поряд із працівником, але не заважати йому;

– ті предмети, якими користуються частіше, розташовуються ближче, ніж ті предмети, якими користуються рідше;

– предмети, які беруть лівою рукою, повинні бути зліва, а ті предмети, які беруть правою рукою – справа;

– якщо використовують обидві руки, то місце розташування пристосувань вибирається з урахуванням зручності захоплювання його двома руками;

– робоче місце не повинно бути захаращене;

– організація робочого місця повинна забезпечувати необхідну оглядовість.

Статичні напруження працівника в процесі праці пов'язані з підтриманням у нерухомому стані предметів і знарядь праці, а також підтриманням робочої пози.

Робоча поза – це основне положення працівника у просторі: зручна робоча поза має забезпечувати стійкість положення корпуса, ніг, рук, голови працівника під час роботи, мінімальні затрати енергії та максимальну результативність праці. Неправильна сидяча поза може викликати застій крові в ногах, а якщо виконується великий обсяг роботи для пальців рук – запалення суглобів.

Організація робочого місця користувача комп'ютера повинна забезпечувати відповідність усіх елементів робочого місця та їх взаємного розташування ергономічним вимогам (рисунок 4.1).

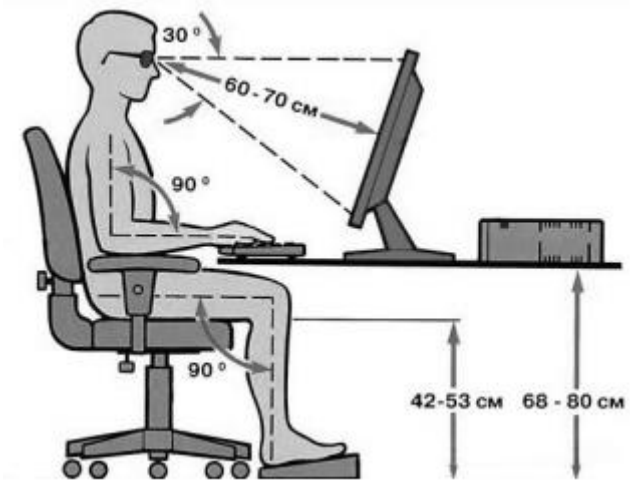

Рисунок 4.1 - Робоче місце і робоча поза користувача ПК

Найпоширенішими у процесі праці є пози сидячи і стоячи. Проектуючи робоче місце, потрібно враховувати, що при виконанні роботи з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи.

Робоча поза стоячи втомлює людину більше, ніж сидяча. Вона вимагає на 10% більше енергії, спричиняє підвищення артеріального і венозного тиску крові, розширення вен на ногах, пошкодження ступень, викривлення хребта [40].

4.3 Висновки до четвертого розділу

В цьому розділі розглянуто важливі питання охорони праці та безпеки в надзвичайних ситуаціях, зокрема описані небезпечні й шкідливі фактори при виконанні робіт за комп’ютером та ергономічні вимоги до робочого місця користувача ПК.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи була спроектована і розроблена РСЗД для хмарної платформи OpenStack.

У процесі виконання роботи були вирішені наступні завдання:

– проаналізовано доступні Linux / Unix OpenSource РСЗД (включаючи розподілені ФС) на прикладі Ceph, GlusterFS, NDFS, HDFS, pNFS;

– за результатами порівняння основних характеристик та параметрів вибрано Ceph, яка найбільш повно задовольняє всім критеріям;

– розроблено і впроваджено скрипт високої доступності для OpenStack;

– спроектовано архітектуру кластера;

– здійснено інтеграцію з хмарною платформою OpenStack.

Так як розподілене сховище даних Ceph може бути розгорнуте на недорогих стандартних серверах, дане рішення є вигідною заміною дорогим сучасним СЗД, при цьому не поступаючись їм в продуктивності.

Планується розширення функціональності OpenStack для реалізації функції OpenStack nova MultiStorage support. Ця функція дозволяє для кожного примірника nova запускати віртуальні машини в різних пулах.

ПЕРЕЛІК ДЖЕРЕЛ

1. Облачные технологи. Теория и практика / Д.Н.Монахов, Н.В.Монахов, Г.Б.Прончев [и др]. М. : МАКС Пресс, МГУ, 2013. 128 с.

2. Горелов А. Куда идут «облака». [Електронний ресурс] – Режим доступу: http://compress.ru/ article.aspx?id =22659#02 (дата звернення 26.11.2020).

3. Хмарні обчислення. [Електронний ресурс] – Режим доступу: http://integritysys.com.ua/solutions/ pricatecloud-solution/ (дата звернення 24.11.2020).

4. Баран І.О., Воронін В.С. До питання розробки системи збереження даних для хмарної платформи OpenStack / Матеріали IX Міжнародної науково-технічної конференції молодих учених та студентів «Актуальні задачі сучасних технологій».– Тернопіль, ТНТУ, 25-26 листопада 2020 р. – Том 2., с. 6.

5. The Most Widely Deployed Open Source Cloud Software in the World [Електронний ресурс] – Режим доступу: https://www.openstack.org (дата звернення: 24.10.2020)

6. Bumgardner Cody K.V., Pipes Jay. OpenStack in Action. Manning, 2016. - 384p.

7. Маркелов А. OpenStack. Знакомство с облачной операционной системой. Четвертое издание. ДМК Пресс, 2018. – 306 с.

8. Облако OpenStack: мифы и реальность [Електронний ресурс] – Режим доступу https://habrahabr.ru/company/jetinfosystems/blog/247087/дата (звернення: 24.10.2020).

9. Mell P., Grance T. The NIST Definition of Cloud Computin. Version 15, 10-7-09. [Електронний ресурс] – Режим доступу: http://csrc.nist.gov/groups/ SNS/cloud - computing/cloud - def- v15.doc. (дата звернення 8.11.2020).

10. Raza M. Public Cloud Growth Trends and the Future Outlook. [Електронний ресурс] – Режим доступу: https://www.bmc.com/blogs/cloud-growth-trends/ (дата звернення 8.11.2020).

11. Риз Дж. Облачные вычисления (CloudApplicationArchitectures) / Дж. Риз. – СПб. : БХВ-Петербург, 2011. – 288 с.

12. Облачные вычисления (мировой рынок). [Електронний ресурс] – Режим доступу: http://www.tadviser.ru/ index.php/Статья:Облачные_вычисления_ %28мировой_рынок%29#Gartner_ (дата звернення 16.11.2020).

13. Manual/LustreManual20 HTML/UnderstandingLustre.html – Lustre Wiki [Електронний ресурс] – Режим доступу: http://wiki.lustre.org/Manual/LustreManual20_HTML/UnderstandingLustre.html (дата звернення 16.11.2020).

14. pNFS & NFSv4.2; a filesystem for grid, virtualization and database by Alex McDonald [Електронний ресурс] – Режим доступу:: http://snia.org/sites/default/education/tutorials/2011/fall/Networking/AlexMcDonald_ pNFS_NFSv43-2.pdf (дата звернення 16.11.2020).

15. Should I use Stripe on GlusterFS? by Joe Julian [Електронний ресурс] – Режим доступу: https://joejulian.name/blog/should-i-use-stripe-on-glusterfs/ (дата звернення 16.11.2020).

16. Large Object Support – swift.2.3.1.dev64 documentation [Електронний ресурс] – Режим доступу: http://docs.openstack.org/developer/swift/ overview_large_objects.html (дата звернення 16.11.2020).

17. Store Cluster Quick Start – Ceph Documentation [Електронний ресурс] – Режим доступу: http://ceph.com/docs/dumpling/start/quick-ceph-deploy/ (дата звернення 16.11.2020).

18. Architecture – Ceph Documentation [Електронний ресурс] – Режим доступу: http://ceph.com/docs/master/architecture (дата звернення: 24.10.2020)

19. Manual Deployment – Ceph Documentation [Електронний ресурс] – Режим доступу: http://ceph.com/docs/master/install/manual-deployment/ (дата звернення: 29.09.2020).

20. Antonopoulos N. Cloud Computing. Principles. Systems and Applications / N. Antonopoulos, L. Gillam. – London; NewYork : Springer-Verlag, 2010. – 379 p.

21. Adkins S., Belamaric J., Giersch V. et al. Openstack Cloud Application Developmentю John Wiley & Sons, 2015. — 168 p.

22. Add/Remove OSDs – Ceph Documentation [Електронний ресурс] – Режим доступу: http://ceph.com/docs/master/rados/deployment/ceph-deploy-osd/, (дата звернення: 29.09.2020)

23. Block Devices And OpenStack – Ceph Documentation [Електронний ресурс] – Режим доступу: http://ceph.com/docs/master/rbd/rbd-openstack/ (дата звернення: 29.09.2020).

24. How to Install OpenStack on CentOS 8 with Packstack [Електронний ресурс] – Режим доступу: https://www.linuxtechi.com/install-openstack-centos-8-with-packstack (дата звернення: 30.11.2020)

25. Evaluating OpenStack: Single-Node Deployment [Електронний ресурс] – Режим доступу: https://access.redhat.com/articles/1127153 (дата звернення: 30.11.2020)

26. Markelov A. Certified OpenStack Administrator Study Guide .Apress, 2016. - 185 p.

27. Cache tiering – Ceph Documentation [Електронний ресурс] – Режим доступу: http://ceph.com/docs/master/rados/operations/cache-tiering (дата звернення: 30.11.2020).

28. Kapadia, A. Implementing cloud storage with Openstack swift: Design, implement, and successfully manage your own cloud storage cluster using the popular OpenStack swift software. / Kapadia, A., Varma, S., Rajana, K. // United Kingdom: Packt Publishing, 2014 – pp. 134 – 136.

29. Arnold, J. OpenStack swift: Using, administering, and developing for swift object storage. / Arnold, J. // O'Reilly Media, 2014 – pp. 254 – 256.

30. Openstack Swift developer documentation. Електронний ресурс] – Режим доступу: http://docs.openstack.org/developer/swift/ (дата звернення: 30.11.2020).

31. Arnold Joe. OpenStack Swiftю O'Reilly Media, 2014. — 338 p.

32. Fifield Tom, Fleming Diane, Gentle Anne ect. OpenStack Operations Guide. O'Reilly Media, 2014. – 330 p.

33. Jackson Kevin. OpenStack Cloud Computing Cookbook. Packt Publishing, 2018. — 398 p.

34. Khedher O. Mastering OpenStack. Packt Publishing, 2015. — 400 p.

35. Distributed storage performance for OpenStack clouds: RedHat storage server vs. Ceph storage by RedHat [Електронний ресурс] – Режим доступу: http://www.principledtechnologies.com/Red%20Hat/RedHatStorage_Ceph_1113.pdf (дата звернення: 30.11.2020).

36. Radez Dan. OpenStack Essentials. Packt Publishing, 2016. — 182 p.

37. Жидецький В. Ц. Основи охорони праці. Львів : Укр. акад. друкарства, 2006. — 336 с.

38. Толок А.О. Крюковська О.А. Безпека життєдіяльності: Навч. посібник. – 2011. – 215 с.

39. Яремко З. М. Безпека життєдіяльності: Навч. посіб. — Львів., 2005. – 301 с.

40. Желібо Є. П. Заверуха Н.М., Зацарний В.В. Безпека життєдіяльності. Навчальний посібник. / Є. Желібо Є.П., Н.М. Заверуха П., В.В. Зацарний. – К.; Каравела, 2004. -328 с.

ДОДАТКИ

ДОДАТОК А

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Маріборский університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедиміната (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том ІІ

ІХ Міжнародної науково-технічної конференції молодих учених та студентів

25-26 листопада 2020 року

УКРАЇНА
ТЕРНОПІЛЬ – 2020

ЗМІСТ

Секція: КОМПЮТЕРНО-ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ЗВ'ЯЗКУ

УДК 004.75

І.О. Баран канд.техн.наук, доц., В.С. Воронін
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ДО ПИТАННЯ РОЗРОБКИ СИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ ДЛЯ ХМАРНОЇ ПЛАТФОРМИ OPENSTACK

I.O. Baran Ph.D., Assoc. Prof., V.S. Voronin

TO A PROBLEM ON DEVELOPMENT OF DATA STORAGE SYSTEM FOR THE OPENSTACK CLOUD PLATFORM

Розподілена система зберігання даних (РСЗД) для хмарної платформи OpenStack призначена для зберігання даних, які доступні для цієї платформи. Під даними, в контексті хмарної платформи, потрібно розуміти наступне: ефемерні диски серверів (ephemeral Nova volumes); постійні диски (Cinder volumes); образи віртуальних серверів (Glance images). РСЗД створюється з метою забезпечення надійного розподіленого зберігання інформації в хмарній платформі. В результаті створення даної системи повинні бути забезпечені такі показники, як масштабованість та відмовостійкість.

В системі виділені наступні функціональні підсистеми: розподілене сховище даних; інтерфейси управління РСЗД та хмарною платформою OpenStack. Забезпечення пристосовуваності системи до змін повинно виконуватися за рахунок: збільшення сумарної ємності системи при додаванні нового сервера зберігання; зменшення сумарної ємності системи при видаленні сервера зберігання (наприклад, в результаті збою сервера); своєчасної заміни обладнання, що вийшло з ладу; модернізації архітектури і інтерфейсу відповідно до нових вимог; своєчасного адміністрування сервера; оперативного реагування на побажання користувачів. Надійність РСЗД повинна забезпечуватися за рахунок: відсутності єдиної точки відмови в архітектурі; розподіленого зберігання даних з автоматичною реплікацією по мережі між серверами; можливості відновлення необхідного рівня відмовостійкості при виході з ладу частини обладнання; можливості зміни рівня відмовостійкості (фактора реплікації) для різних пулів. Система повинна надавати інтерфейс командного рядка для управління. Інформаційна безпека в системі повинна здійснюватися за рахунок доступу до управління по захищених протоколах SSH / HTTPS та аутентифікації РСЗД всередині кластера (між серверами). Вимоги до функцій, які має виконувати система: використання тільки open-source технологій; підтримувані ОС - Red Hat Enterprise Linux, CentOS, Fedora; як база даних (якщо потрібно для Web-інтерфейсу) повинна використовуватися СУБД MySQL або MariaDB; автоматична реплікація даних між серверами; можливість відновлення даних після збою частини обладнання; можливість задання особливих параметрів реплікації даних з урахуванням можливих failure доменів; природна інтеграція з OpenStack (nova, cinder, glance і swift); можливість використання моментальних знімків (снепшот); можливість використання твердотільних накопичувачів для більш вимогливих клієнтів до продуктивності системи. Для якісного виконання завдання необхідно зробити наступне: провести аналіз доступних Linux / Unix OpenSource РСЗД (включаючи розподілені файлові системи) на прикладі Ceph, GlusterFS, LustreFS, Swift, pNFS; здійснити вибір РСЗД, яка найповніше задовольняє всім критеріям; вибрати найкращі сценарії використання обраних РСЗД; спроектувати архітектуру; оптимізувати продуктивність; здійснити інтеграцію з хмарною платформою OpenStack. Так як розподілене сховище даних Ceph може бути розгорнутим на недорогих стандартних серверах, дане рішення буде вигідною заміною дорогим сучасним системам зберігання даних, при цьому не поступаючись їм в продуктивності.

ДОДАТОК Б
Лістинг скрипта HA для OpenStack

```
#! / Bin / bash
script_status () {
pid_file = "/ var / run / HA_nova.pid"
if [-a $ pid_file]
then
log_w "script $ 0 is running"
exit 0
else
echo $! > $ Pid_file
check_sr_status
fi
}
log_w () {
log_file = "/ var / log / HA_nova.log"
if [-a $ log_file]
then
echo `date + '% b% e% R': $ 0:` "$ @" >> $ log_file
else  touch $ log_file
echo "creation date is" `date` >> $ log_file
echo `date + '% b% e% R': $ 0:` "$ @" >> $ log_file
fi
}
check_sr_status () {
sr_status = $ (nova service-list | grep "nova-compute" | awk
'{print $ 12 "" $ 6}')
echo "$ sr_status" | while read V1 V2
do
if [ "$ V1"! = "up"]
then  log_w "failed server is: $ V2 status $ V1" migrating
else
log_w "All looks good, nothing migrate here, have a nice work"
log_w "$ V2 status is $ V1"
fi
done
}
migrating () {
failed_sr_name = $ (nova service-list | grep "nova-compute" |
grep -v "up" | awk
'{Print $ 6}')
echo "$ failed_sr_name" | while read V3
do
nova    host-evacuate    --on-shared-storage    $    V3    >>
/var/log/HA_nova.log
done
}
script_status
rm -rf $ pid_file
exit 0
```