НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут телекомунікаційних систем
Кафедра Телекомунікаційних систем

До захисту допущено:

Завідувач кафедри

_____Леонід УРИВСЬКИЙ

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Телекомунікаційні системи та мережі»
спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Побудова комп'ютерних кластерів для моделювання надскладних процесів у телекомунікаційних мережах на базі операційної системи Linux»

Виконала:
студентка IV курсу, групи ТС-71
Голосова Вікторія Денисівна

_____

Керівник:
Ст. викладач кафедри ТС
Вакуленко Олександр Володимирович

_____

Рецензент:
доцент кафедри ТК, к.т.н., доцент
Явіся Валерій Сергійович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студентка _____

Київ – 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Телекомунікаційних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Телекомунікаційні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Леонід УРИВСЬКИЙ

«___»_____20__ р.

ЗАВДАННЯ

на дипломну роботу студенту

Голосовій Вікторії Денисівні

1. Тема роботи «Побудова комп'ютерних кластерів для моделювання надскладних процесів у телекомунікаційних мережах на базі операційної системи Linux», керівник роботи Вакуленко Олександр Володимирович, старший викладач, затверджені наказом по університету від «14» квітня 2021 р. №1007-с

2. Термін подання студентом роботи 9 червня 2021 року.

3. Вихідні дані до роботи: Робочі станції закладу вищої освіти; програмне забезпечення для ОС Linux

4. Зміст роботи(дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити): 1. Аналіз сучасного стану застосування комп'ютерних кластерів для моделювання надскладних процесів; 2. Загальна характеристика операційної системи Linux; 3. Порядок побудови комп'ютерних кластерів для моделювання надскладних процеів на базі операційної системи Linux.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) мультимедійна презентація в Power Point.

6. Дата видачі завдання 11.09.2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи.	Термін виконання етапів роботи	Примітка
1	Ознайомлення з літературою.	01.03.2021 – 11.04.2021	Вик.
2	Підготовка першого розділу «Аналіз сучасного стану застосування комп'ютерних кластерів ддля моделювання надскладних процесів».	12.04.2021 – 30.04.2021	Вик.
3	Підготовка другого розділу «Загальна характеристики операційної системи Linux».	01.05.2021 – 14.05.2021	Вик.
4	Підготовка третього розділу «Порядок побудови комп’ютерних кластерів для моделювання надскладних прооцесів на базі операційної системи Linux».	15.05.2021 – 03.06.2021	Вик.
5	Виготовлення ілюстративного матеріалу.	04.06.2021 – 06.06.2021	Вик.
6	Підготовка загального висновку.	07.06.2021	Вик.
7	Остаточне оформлення роботи	08.06.2021	Вик.
8	Подання роботи в бібліотеку	11.06.021	Вик.

Студент: Вікторія ГОЛОСОВА

Керівник: Олександр ВАКУЛЕНКО

РЕФЕРАТ

Дипломна робота: 68 с., 19 рис., 1 табл., 12 джерел.

Дана дипломна робота розкриває аналіз сучасного стану застовування комп’ютерних кластерів для моделювання надскладних процесів та містить результати дослідженння принципи побудови комп’ютерних кластерів, а також розроблено порядок побудови комп’ютерних кластерів на базі операційної системи Linux.

Об`єкт дослідження – процеси функціонування комп’ютерних кластерів під усправлінням операційної системи Linux.

Метою роботи є підвищення ефективності навчалного процесу при проведенні досліджень шляхом моделювання надскладних процесів у телекомунікаційних системах на базі комп’ютерних кластерів під управлінням операційної системи Linux.

Побудова комп’ютерного кластера для моделювання надскладних процесів у телекомунікаційних системах на базі операційної системи Linux надасть можливість підвищити ефективність навчального процесу та розширити коло компетентностей студентів, які навчаються за спеціальністю 172 Телекомунікації та радіотехніка.

ВІРТУАЛЬНА МЕРЕЖА, MPI, КАНАЛ ЗВ’ЯЗКУ, МЕТОД, INTERNET, КОМУТАТОР, КЛАСТЕР, ОС LINUX

ABSTRACT

Diploma thesis: 68 p., 19 fig., 1 tables., 12 refer.

This thesis reveals an analysis of the current state of computer clusters for modeling super-complex processes and contains the results of the study of the principles of computer clusters, as well as the procedure for building computer clusters based on the Linux operating system.

The object of research is the processes of functioning of computer clusters under the control of the Linux operating system.

The aim of the work is to increase the efficiency of the educational process in conducting research by modeling extremely complex processes in telecommunications systems based on computer clusters running the Linux operating system.

Building a computer cluster for modeling complex processes in telecommunications systems based on the Linux operating system will provide an opportunity to increase the efficiency of the educational process and expand the range of competencies of students majoring in 172 Telecommunications and Radio Engineering.

VITUAL NETWORK, MPI, COMMUNICATION CHANNEL, METHOD, INTERNET, SWITCH, CLUSTER, LINUX OS

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ARM – Advanced RISC Machines

BMC – Baseboard Management Controller

DEB – Debian

GNU – General Public License

HA – High Availability

HPC Cluster – High-Performance Computing Cluster

KDE – K Desktop Environment

LAM – Lobe Attachment Module

LXDE – Lightweight X11 Desktop Environment

MPI – Message Passing Interface

MPICH – Message Passing Interface CHameleon

RAID – Redundant Array of Independent Disks

RISC-V – Reduced Instruction Set Computer

SCALAPACK – Scalable Linear Algebra Package

SMP – Symmetric Multiprocessing

SPMD – Single Program, Multiple Data

ОС – Операційна Система

ПЗ – Програмне Забезпечення

ВСТУП

Актуальність теми. Проблемам побудови комп'ютерних кластерів для моделювання надскладних процесів у телекомунікаційних системах на базі операційної системи Linux присвячено останніми роками багато уваги [1-8]. Одним з напрямів, являється організація кластерів на базі вже існуючих мереж робочих станцій, тобто робочі станції можуть використовуватися в якості вузлів кластера вночі та в неробочі дні. Системи такого типу називають CoW (Cluster of Workstations). В цьому випадку реальним видається варіант, коли кластер будується на основі існуючого комп'ютерного класу та окремих робочих станцій закладу вищої освіти. Під час навчального процесу виникають задачі пов’язані з проведенням моделювання надскладних процесів у телекомунікаційних системах для яких продуктивність одного комп’ютера недостатньо. Тому побудова і застосування комп’ютерних кластерів на базі робочих станцій закладу вищої освіти являється актуальним в наш час.

Новизна отриманих результатів визначена в комплексному підході щодо побудови комп’ютерних кластерів для моделювання надскладних процесів у телекомунікаційних системах на базі операційної системи Linux.

Практичне значення одержаних результатів. Результати дослідження доцільно використовувати під час навчального процесу для проведення наукових досліджень шляхом моделювання надскадних процесів у телекомунікаційних системах на базі комп’ютерних кластерів побудованих на робочих станціях закладу вищої освіти.

Об`єкт дослідження*:* процеси функціонування комп’ютерних кластерів під управлінням операційної системи Linux.

Предмет дослідження*:* методики побудови комп’ютерних кластерів для моделювання надскладних процесів в телекомунікаційних системах на

базі операційної системи Linux.

Мета роботи*:* підвищення ефективності начального процесу при проведенні досліджень шляхом моделювання надскладних процесів в телекомунікаційних системі на базі операційної системи Linux.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ЗАСТОСУВАННЯ КОМП’ЮТЕРНИХ КЛАСТЕРІВ ДЛЯ МОДЕЛЮВАННЯ НАДСКЛАДНИХ ПРОЦЕСІВ

1.1 Загальні відомості про обчислювальний кластер

Комп’ютерний кластер – це шар серверів, які підключені до певної комунікаційної мережі. Кожен комп'ютерний вузол має власну оперативну пам'ять і запускає власну операційну систему.

Найпоширенішим є використання змішаних стовпців, тобто стовпців, в яких всі вузли абсолютно однакові з точки зору архітектури та функції.

Для кожного стовпця є виділений сервер – вузол управління (передній). На комп’ютері встановлено програмне забезпечення, яке активує вузли комп’ютера під час запуску системи та керує запуском програми в кластері.

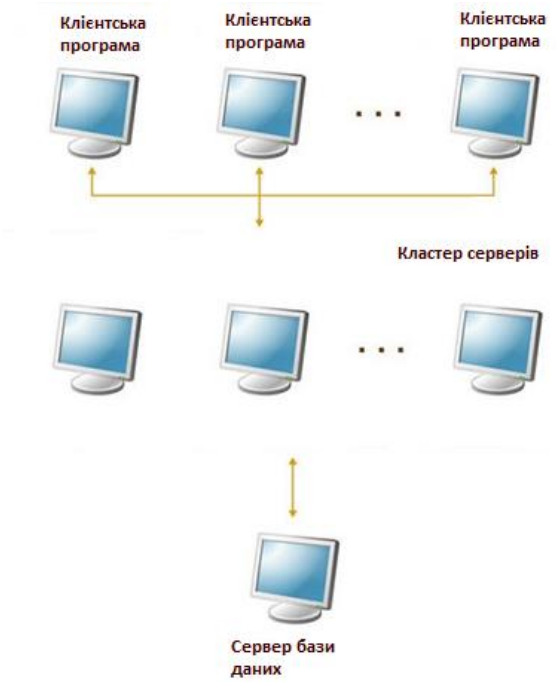

Рисунок 1.1 – Обчислювальний кластер

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

Користувачі мають домашні каталоги на сервері доступу - шлюзі (цей сервер забезпечує зв'язок кластера із зовнішнім світом через корпоративну ЛОМ або Інтернет), безпосередній доступ користувачів на керуючий вузол виключається, а доступ на обчислювальні вузли кластера можливий (наприклад, для ручного керування компіляцією завдання).

Обчислювальний кластер, як правило, працює під керуванням однієї з різновидів ОС Unix – багатокористувацької багатозадачної мережевої операційної системи.

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

Існує декілька способів зайняти обчислювальні потужності кластера:

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

Запускати готові паралельні програми. Для деяких завдань доступні безкоштовні або комерційні паралельні програми, які при необхідності Ви можете використовувати на кластері. Як правило, для цього досить, щоб програма була доступна у вихідних текстах, реалізована з використанням інтерфейсу MPI на мовах C/C++ або Фортран.

Викликати у своїх програмах паралельні бібліотеки. Для деяких областей, наприклад, лінійна алгебра, доступні бібліотеки, які дозволяють вирішувати широке коло стандартних підзадач з використанням можливостей паралельної обробки. Якщо звертання до таких підзадач

становить більшу частину обчислювальних операцій програми, то використання такої паралельної бібліотеки дозволить одержати паралельну програму практично без написання власного паралельного коду. Прикладом такої бібліотеки є SCALAPACK, яка доступна для використання на нашому кластері.

Створювати власні паралельні програми. Це найбільш трудомісткий, але й найбільш універсальний спосіб. Існує кілька варіантів такої роботи, зокрема , вставляти паралельні конструкції в готові паралельні програми або створювати з "нуля" паралельну програму.

Паралельні програми на обчислювальному кластері працюють у моделі передачі повідомлень – message passing. Це значить, що програма складається з багатьох процесів, кожний з яких працює на своєму процесорі й має свій адресний простір.

Причому безпосередній доступ до пам'яті іншого процесу неможливий, а обмін даними між процесами відбувається за допомогою операцій приймання й посилки повідомлень.

Тобто процес, який повинен одержати дані, викликає операцію receive (прийняти повідомлення), і вказує, від якого саме процесу він повинен одержати дані, а процес, який повинен передати дані іншому, викликає операцію send (послати повідомлення) і вказує, якому саме процесу потрібно передати ці дані.

Ця модель реалізована за допомогою стандартного інтерфейсу MPI. Існує кілька реалізацій MPI, у тому числі безкоштовні й комерційні, переміщені й орієнтовані на конкретну комунікаційну мережу. На кластерах СКІТ-1 і СКІТ-2 ми використовуємо комерційну реалізацію Scali (для интерконнекта SCI) і безкоштовну Openmpi (для интерконнекта Infiniband).

Як правило, Mpi – програми побудовані по моделі SPMD (одна програма – багато даних), тобто для всіх процесів є тільки один код

програми, а різні процеси зберігають різні дані й виконують свої дії залежно від порядкового номера процесу.

Для прискорення роботи паралельних програм варто прийняти заходи для зниження накладних витрат на синхронізацію при обміні даними. Можливо, прийнятним підходом виявиться сполучення асинхронних пересилань і обчислень.

Для виключення простою окремих процесорів потрібно найбільш рівномірно розподілити обчислення між процесами, причому в деяких випадках може знадобитися динамічне балансування. Важливим показником, який говорить про те, чи ефективно в програмі реалізований паралелізм, є завантаження обчислювальних вузлів, на яких працює програма.

Якщо завантаження на всіх або на частині вузлів далека від 100 % – виходить, програма неефективно використовує обчислювальні ресурси, тобто створює більші накладні витрати на обміни даними або нерівномірно розподіляє обчислення між процесами. Користувачі 1 і 2 можуть подивитися завантаження через веб-інтерфейс для перегляду стану вузлів.

В деяких випадках для розуміння, в чому причина низької продуктивності програми і які саме місця в програмі необхідно модифікувати, щоб добитися збільшення продуктивності, має сенс використовувати спеціальні засоби аналізу продуктивності – профілірovщики і трасувальники [1].

Переваги кластерної системи перед набором незалежних комп'ютерів очевидні. По-перше, розроблено безліч диспетчерських систем пакетної обробки завдань, що дозволяють послати завдання на обробку кластеру в цілому, а не якогось окремого комп'ютера. Ці диспетчерські системи автоматично розподіляють завдання за вільними обчислювальним вузлам або буферизують їх при відсутності таких, що дозволяє забезпечити більш рівномірну і ефективне завантаження

комп'ютерів. По-друге, з'являється можливість спільного використання обчислювальних ресурсів декількох комп'ютерів для вирішення одного завдання.

Для створення кластерів зазвичай використовуються або прості однопроцесорні персональні комп'ютери, або двох- або чотирьох-процесорні SMP-сервери. При цьому не накладається ніяких обмежень на склад і архітектуру вузлів. Кожен з вузлів може функціонувати під управлінням своєї власної операційної системи. Найчастіше використовуються стандартні ОС: Linux, FreeBSD, Solaris, Tru64 Unix, Windows NT. У тих випадках, коли вузли кластера неоднорідні, то говорять про гетерогенних кластерах.

При створенні кластерів можна виділити два підходи. Перший підхід застосовується при створенні невеликих кластерних систем. У кластер об'єднуються повнофункціональні комп'ютери, які продовжують працювати і як самостійні одиниці, наприклад, комп'ютери навчального класу або робочі станції лабораторії. Другий підхід застосовується в тих випадках, коли цілеспрямовано створюється потужний обчислювальний ресурс. Тоді системні блоки комп'ютерів компактно розміщуються в спеціальних стійках, а для управління системою і для запуску завдань виділяється один або декілька повнофункціональних комп'ютерів, званих хост-комп'ютерами. У цьому випадку немає необхідності постачати комп'ютери обчислювальних вузлів графічними картами, моніторами, дисковими накопичувачами і іншим периферійним обладнанням, що значно здешевлює вартість системи [2].

1.2 Класифікація кластерів

1) Кластери високої доступності

Позначаються абревіатурою HA (англ. High Availability – висока доступність). Створюються для забезпечення високої доступності сервісу,

що надається кластером. Надмірна кількість вузлів, що входять в кластер, гарантує надання сервісу у разі відмови одного або декількох серверів. Типове число вузлів – два, це мінімальна кількість, що приводить до підвищення доступності. Створено безліч програмних рішень для побудови такого роду кластерів.

Відмовостійкі кластери та системи взагалі будуються по трьом основним принципам:

З холодним резервом або активний / пасивний. Активний вузол виконує запити, а пасивний чекає його відмови і включається в роботу, коли така відбудеться. Приклад – резервні мережеві з'єднання, зокрема, алгоритм зв'язуючого дерева.

З гарячим резервом або активний / активний. Всі вузли виконують запити, в разі відмови одного навантаження перерозподіляється між рештою. Тобто кластер розподілення навантаження з підтримкою перерозподілу запитів при відмові. Прикладами є практично всі кластерні технології, наприклад, Microsoft Cluster Server. OpenSource проект OpenMosix.

З модульною надмірністю. Застосовується тільки у випадку, коли простій системи абсолютно неприпустимий. Всі вузли одночасно виконують один і той же запит (або частини його, але так, що результат досяжний і при відмові будь-якого вузла), з результатів береться будь-який. Необхідно гарантувати, що результати різних вузлів завжди будуть однакові (або відмінності гарантовано не вплинуть на подальшу роботу). Приклади - RAID та Triple modular redundancy.

Конкретна технологія може поєднувати дані принципи в будь-якій комбінації. Наприклад, Linux-HA підтримує режим обопільної поглинаючої конфігурації (англ. takeover), в якому критичні запити виконуються усіма вузлами разом, інші ж рівномірно розподіляються між ними.

2) Кластери розподілу навантаження

Принцип їх дії будується на розподілі запитів через один або кілька вхідних вузлів, які перенаправляють їх на обробку в інші, обчислювальні вузли. Початкова мета такого кластера - продуктивність, однак, у них часто використовуються також і методи, що підвищують надійність. Подібні конструкції називаються серверними фермами. Програмне забезпечення (ПЗ) може бути як комерційним так і безкоштовним.

3) Обчислювальні кластери

Кластери використовуються в обчислювальних цілях, зокрема в наукових дослідженнях. Для обчислювальних кластерів вагомими показниками є висока продуктивність процесора в операціях над числами з рухомою комою (flops) і низька латентність об'єднувальної мережі, і менш вагомими - швидкість операцій введення-виведення, яка більшою мірою важлива для баз даних та web-сервісів. Обчислювальні кластери дозволяють зменшити час розрахунків, порівняно з одиночним комп'ютером, розбиваючи завдання на паралельно виконувані гілки, які обмінюються даними через мережу. Одна з типових конфігурацій - набір комп'ютерів, зібраних із загальнодоступних компонентів, з встановленою на них операційною системою Linux, і пов'язаних мережею Ethernet, Myrinet, InfiniBand або іншими відносно недорогими мережами. Таку систему прийнято називати кластером Beowulf.

Окремо виділяють високопродуктивні кластери (позначаються англ. абревіатурою HPC Cluster - High-performance computing cluster). Список найпотужніших високопродуктивних комп'ютерів (також може позначатися англ. абревіатурою HPC) можна знайти в світовому рейтингу ТОП500.

4) Системи розподілених обчислень (grid)

Такі системи не прийнято вважати кластерами, але їх принципи в значній мірі схожі з кластерною технологією. Їх також називають grid-

системами. Головна відмінність - низька доступність кожного вузла, тобто неможливість гарантувати його роботу в заданий момент часу (вузли підключаються і відключаються в процесі роботи), тому завдання повинне бути розбите на ряд незалежних один від одного процесів. Така система, на відміну від кластерів, не схожа на єдиний комп'ютер, а служить спрощеним засобом розподілу обчислень. Нестабільність конфігурації, в такому випадку, компенсується великим числом вузлів.

5) Кластер серверів, організованих програмно

Кластер серверів (в інформаційних технологіях) - група серверів, об'єднаних логічно, здатних обробляти ідентичні запит и і використовуються як єдиний ресурс. Найчастіше сервери групуються за допомогою локальної мережі. Група серверів володіє більшою надійністю і більшою продуктивністю, ніж один сервер. Об'єднання серверів в один ресурс відбувається на рівні програмних протоколів.

На відміну від апаратного кластера комп'ютерів, кластери організовувані програмно, вимагають:

Наявності спеціального програмного модуля (Cluster Manager), основною функцією якого є підтримка взаємодії між усіма серверами - членами кластеру:

Синхронізації даних між усіма серверами - членами кластеру;

розподіл навантаження (клієнтських запитів) між серверами – членами кластеру;

від уміння клієнтського програмного забезпечення розпізнавати сервер, що являє собою кластер серверів, і відповідним чином обробляти команди від Cluster Manager.

Якщо клієнтська програма не вміє розпізнавати кластер, вона буде працювати тільки з тим сервером, до якого звернулася спочатку, а при спробі Cluster Manager перерозподілити запит на інші сервери, клієнтська

програма може взагалі позбутися доступу до цього сервера (результат залежить від конкретної реалізації кластера) [3].

1.3 Аналіз застосування паралельних програм на комп’ютерному кластері

Паралельні програми на обчислювальному кластері працюють в моделі передачі повідомлень (message passing). Це означає, що програма складається з безлічі процесів, кожен з яких працює у своєму процесорі і має особистий адресний простір. Причому безпосередній доступ до пам'яті іншого процесу не є можливий, а обмін даними між процесами відбувається за допомогою операцій прийому і посилки повідомлень. Тобто процес, котрий повинен отримати дані, викликає операцію Receive (прийняти повідомлення), і вказує, від якого саме процесу він повинен отримати дані, а процес, який повинен передати дані іншому, викликає операцію Send (надіслати повідомлення) і вказує, котрому саме процесу потрібно передати ці дані. Ця модель реалізована за допомогою стандартного інтерфейсу MPI. Існує декілька реалізацій MPI, в тому числі безкоштовні і комерційні, переносяться і орієнтовані на конкретну комунікаційну мережу.

Як правило, MPI – програми побудовані за моделлю SPMD (одна програма - багато даних), тобто для усіх процесів є один код програми, а різні процеси зберігають дані різні і виконують свої дії в залежності від порядкового номеру процесу.

Для прискорення роботи паралельних програм варто прийняти заходи для зниження накладних витрат при обміні даними на сихронізацію. Можливо прийнятним підходом виявиться сполучення асинхронних пересилань і обчислень.

Для виключення простою окремих процесорів потрібно більш рівномірно розподілити обчислення між процесами, причому в деяких випадках можливо знадобитися динамічне балансування. Дуже важливим показником, який говорить про те, чи ефективно у програмі реалізований паралелізм, є завантаження обчислювальних вузлів, на яких і працює програма.

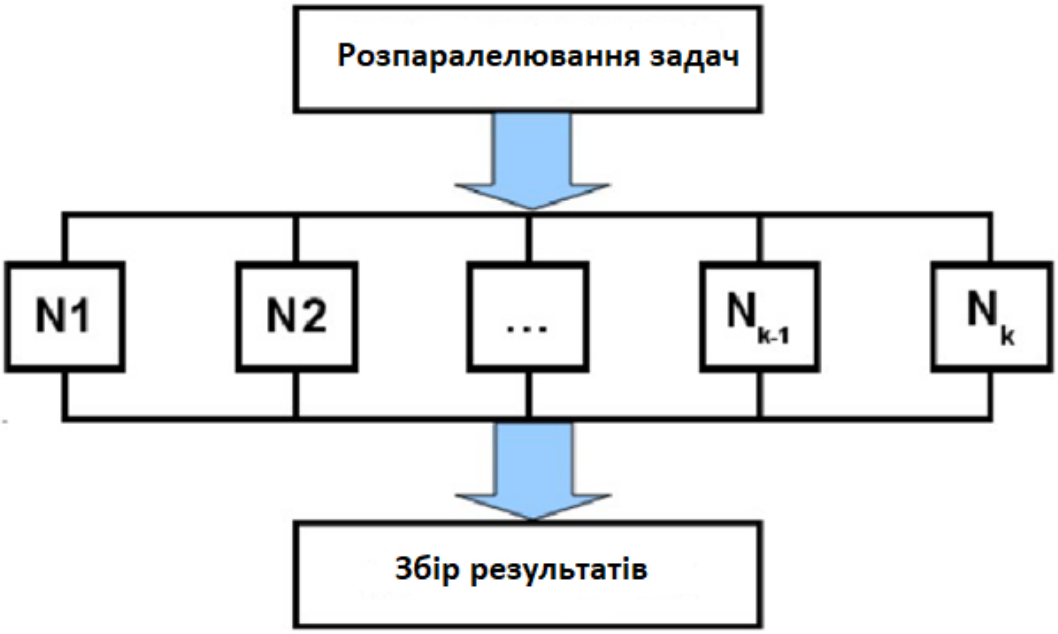

Рисунок 1.2 – Розпаралелювання задач

Якщо завантаження на всіх або на частині вузлів далека від ста відсотків - виходить, програма неефективно використовує обчислювальні ресурси, тобто створює більш накладні витрати на обміни даними або розподіляє обчислення між процесами нерівномірно. Користувачі 1 і 2 для перегляду стану вузлів можуть подивитися завантаження через веб-інтерфейс.

В деяких випадках для розуміння, в чому причина більш низької продуктивності програми і які саме місця в програмі необхідно видозмінювати, щоб домогтися збільшення продуктивності, має сенс

використовувати спеціальні засоби аналізу продуктивності – профіліровщики і трасувальники [1].

1.4 Message Passing Interface

MPI розшифровується як "Message passing interface" ("Інтерфейс передачі повідомлень"). MPI – це стандарт на програмний інструментарій для забезпечення зв'язку між окремими процесами паралельного завдання. MPI надає програмісту єдиний механізм взаємодії процесів всередині паралельно виконується завдання незалежно від машинної архітектури (однопроцесорні, багатопроцесорні із загальною або роздільною пам'яттю), взаємного розташування процесів (на одному фізичному процесорі або на різних) і API операційної системи. Програма, що використовує MPI, легко налагоджували і переноситься на інші платформи, часто для цього досить простий перекомпіляції вихідного тексту програми.

Рішення використовувати в своїх завданнях MPI слід обережно, після ретельного зважування своїх сил і можливостей як програміста. Незважаючи на те, що MPI є значним кроком вперед в порівнянні з попереднім поколінням бібліотек передачі повідомлень, а, можливо і внаслідок цього, програмувати на MPI досить складно. Причиною тому явлется не недолік стандарту, а в самій ідеології передачі повідомлень. MPI можна розглядати як рівень асемблера для паралельних програм.

Основна відмінність стандарту MPI від його попередників (p4, PVM) – поняття комунікатора. Всі операції синхронізації і передачі повідомлень локалізуються всередині комунікатора. З комунікатором зв'язується група процесів. Зокрема, всі колективні операції викликаються одночасно на всіх процесах, що входять в цю групу. Оскільки взаємодія між процесами инкапсулируется всередині комунікатора, на базі MPI можна створювати бібліотеки паралельних програм.

В даний час різними колективами розробників написано кілька програмних пакетів, які відповідають специфікації MPI, зокрема: MPICH, LAM, OpenMPI і так далі. У двох словах охарактеризуємо найпоширеніші з цих пакетів. Якщо говорити про LAM, то основна перевага цього пакету – багаті налагоджувальні можливості. Трасування звернень до MPI і аналіз стану паралельної програми після аварійного завершення роблять процес налагодження менш важким і більш продуктивним. З іншого боку, пакет MPICH більш мобільний, слідуючи простим інструкціям можна перенести MPI на нову платформу (наприклад з Linux на Windows або навпаки). Для цього буде потрібно необхідно написати лише кілька драйверів нижнього рівня. Установка бібліотеки MPICH проходить кілька важче, ніж установка LAM MPI, оскільки доводиться ставити набагато більше число параметрів, причому призначення деяких з них відомо тільки розробникам. Якщо ж говорити про ефективність, то є думка, що MPICH кілька ефективніше передає повідомлення. Сперечатися з цим не буду, але мені особисто переконатися в цьому не довелося. Що стосується налагодження, то налагоджувати програми в середовищі MPICH важче.

MPI – це добре стандартизований механізм для побудови програм по моделі обміну повідомленнями. Існують стандартні "прив'язки" MPI до мов C / C ++, Fortran 77/90. Є і безкоштовні і комерційні реалізації майже для всіх суперкомп'ютерних платформ, а також для High Performance кластерних систем, побудованих на вузлах з ОС Unix, Linux і Windows. В даний час MPI – найбільш широко використовуваний і динамічно розвивається інтерфейс зі свого класу. За стандартизацію MPI відповідає MPI Forum. У новій версії стандарту 2.0 описано велике число нових цікавих механізмів і процедур для організації функціонування паралельних програм: динамічне управління процесами, односторонні комунікації (Put / Get), паралельні I / O. Але на жаль, поки немає повних готових реалізацій

цієї версії стандарту, хоча частина з нововведень вже активно використовується.

Основні поняття MPI. парадигма SPMD

При запуску завдання створюється група з P процесів. Група ідентифікується цілочисельним дескриптором (комунікатором). Усередині групи процеси нумеруються від 0 до P-1. В результаті виконання завдання вихідна група (їй присвоєно ім'я MPI_COMM_WORLD) може ділитися на підгрупи, підгрупи можуть об'єднуватися в нову групу, що має свій комунікатор. Таким чином, процес має можливість одночасно належати кільком групам процесів. Кожному процесу доступний свій номер myProc всередині будь-якої групи, членом якої він є.

Поведінка всіх процесів описується однією і тією ж програмою. Межпроцессной комунікації в ній програмуються явно з використанням бібліотеки MPI, яка і диктує стандарт програмування. Квазіодновременний запуск вихідної групи процесів проводиться засобами операційної системи. При цьому P визначається бажанням користувача, а аж ніяк не кількістю доступних процесорів.

Отже, все P процесів асинхронно виконують одну і ту ж програму. Але у кожного з них свій номер myProc. Тому в програмі, природно, будуть такі фрагменти:

```
if (myProc.eq.0) then
   <Робити щось одне>
else if (myProc.eq.1) then
   <Робити щось інше>
        . . .
else
   <Робити щось P-е>
endif
```

Таким чином, в програмі "під одним дахом" закодовано поведінку всіх процесів. У цьому й полягає парадигма програмування Single Program - Multiple Data (SPMD).

Зазвичай поведінка процесів однаково для всіх, крім одного, який виконує координуючі функції, не відмовляючись, втім, взяти на себе і частину загальної роботи. В якості координатора зазвичай вибирають процес з номером 0 [4].

1.5 Висновки до розділу 1

Кластерні обчислення надають ряд переваг щодо звичайних паралельних комп'ютерів, виготовлених на замовлення, для досягнення продуктивності, більшої, ніж типова для уніпроцесорів. Як наслідок, поява кластерів значно розширила доступність високопродуктивної обробки для набагато ширшого співтовариства та посилила її вплив завдяки новим можливостям в галузі науки, техніки, промисловості, медицини, комерції, фінансів, оборони та освіти серед інших секторів. обчислювального застосування. Серед найважливіших переваг кластерних обчислень є:

– масштабованість продуктивності. Кластеризація комп'ютерних вузлів забезпечує засоби збирання більших систем, ніж це практично для нестандартних паралельних систем, оскільки вони самі можуть стати вузлами кластерів;

– продуктивність за вартістю. Кластеризація масових комп’ютерних систем дає перевагу ринку набагато ширшу, ніж обмежена для високопродуктивного обчислювального співтовариства. Для багатьох застосувань досягається перевага в ціновому відношенні в порівнянні із спеціально розробленими паралельними комп’ютерами;

– гнучкість конфігурації. Організація кластерних систем визначається топологією їх мереж взаємозв'язку, яка може бути визначена

під час встановлення та легко модифікована. Залежно від вимог користувацьких програм, можуть бути реалізовані різні конфігурації системи для оптимізації пропускної здатності та затримки потоку даних;

– простота оновлення. Старі компоненти можуть бути замінені або нові елементи додані до оригінального кластера, щоб поступово покращити роботу системи, зберігаючи більшу частину початкових вкладень в апаратне та програмне забезпечення.

– конвергенція архітектури. Кластерні обчислення пропонують єдину загальну стратегію впровадження та застосування паралельних високопродуктивних систем, незалежних від конкретних постачальників обладнання та їх рішень щодо продукту. Користувачі кластерів можуть будувати системи програмних додатків із впевненістю, що такі системи будуть доступні для їх підтримки в довгостроковій перспективі;

– відстеження технологій. Кластери забезпечують найшвидший шлях до інтеграції найновіших технологій високопродуктивних обчислень, оскільки досягнення технологій пристроїв зазвичай спочатку включаються в комп'ютери масового ринку, придатні для кластеризації;

– висока доступність. Кластери надають кілька зайвих однакових ресурсів, які при правильному управлінні можуть забезпечити безперервну роботу системи завдяки витонченій деградації, навіть якщо окремі компоненти виходять з ладу [5].

2 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ОПЕРАЦІЙНОЇ СИСТМИ LINUX

2.1 Загальні відомості про моделювання

Комп'ютерна модель – це алгоритми і рівняння, що використовуються для захоплення поведінки модельованої системи. Навпаки, комп'ютерне моделювання є фактичним ходом програми, яка містить ці рівняння або алгоритми. Моделювання, отже, є процесом запуску моделі. Таким чином, спершу потрібно «побудувати модель», а потім або «запустити модель» або, що еквівалентно «запустити моделювання».

Вимоги до вхідних даних в моделюванні відрізняються один від одного. Для деяких моделей, вхідні дані можуть складатись з кількох цифр (наприклад, моделювання сигналу змінного електрики на дроті), в той час як інші можуть зажадати терабайт інформації

До основних етапів комп'ютерного моделювання відносяться:

- розробка концептуальної моделі, виявлення основних елементів системи і елементарних актів взаємодії;
- формалізація, тобто перехід до математичної моделі; створення алгоритму та написання програми;
- планування і проведення комп'ютерних експериментів;
- аналіз та інтерпретація результатів.

Розрізняють аналітичне та імітаційне моделювання. При аналітичному моделюванні вивчаються математичні (абстрактні) моделі реального об'єкта у вигляді алгебраїчних, диференціальних та інших рівнянь, а також передбачають здійснення однозначної обчислювальної процедури, що призводить до їх точного розв'язання. При імітаційному моделюванні досліджуються математичні моделі у вигляді алгоритму, що відтворює функціонування досліджуваної системи шляхом послідовного виконання великої кількості елементарних операцій [6].

2.2 Характеристика операційної системи Linux

Linux – це сімейство Unix-подібних операційних систем, які використовують ядро Linux, яке розробив фіно-американський програміст Лінус Торвальдс. Операційні системи, що використовують ядро Linux, називаються збірками Linux, і вони є такими ж самими операційними системами як Microsoft Windows або Apple macOS, але мають одну дуже важливу особливість , а саме – їх вихідні коди є відкритими, так як вони поширюються під ліцензією GNU GPL, яка має на увазі створення безкоштовного і відкритого програмного забезпечення (open source software). Це означає, що у будь-який користувач має право вивчати і змінювати вихідний код.

Архітектура Linux-систем

На наступному рисунку 2.1 показана архітектура Linux-систем:

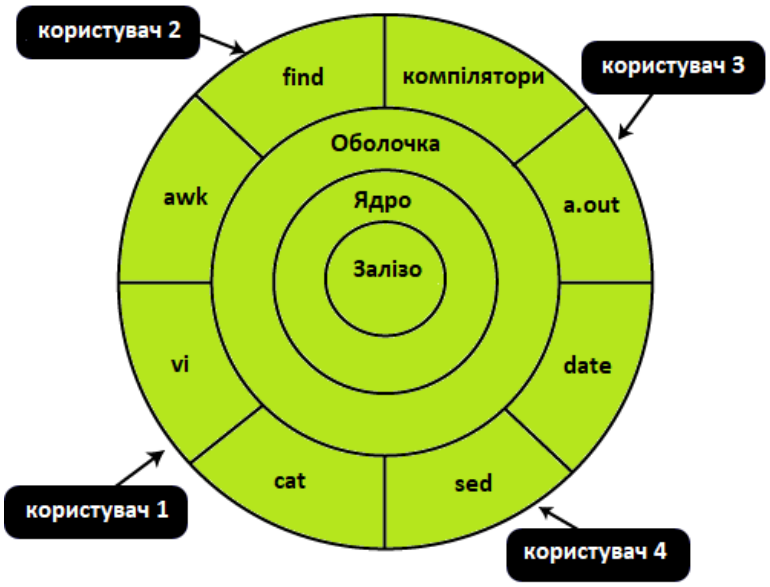

Рисунок 2.1 – Архітектура Linux-систем

«Залізо» – апаратне забезпечення комп'ютера (процесор, відеокарта, оперативна пам'ять і ін.) З усіма його периферійними пристроями.

Ядро – є основним компонентом операційної системи, взаємодіє безпосередньо з апаратним забезпеченням, граючи роль посередника між низькорівневим «залізом» і компонентами верхнього рівня.

Оболонка (або «shell», «командний інтерпретатор») – інтерфейс для взаємодії між користувачами системи і ядром ОС, абстрагується внутрішній устрій системи. Приймає команди від користувачів і запускає на виконання відповідних функцій.

Утиліти (vi, cat, sed, date, компілятори і ін.) – службові програми, які надають користувачеві більшу частину функціональних можливостей операційної системи.

ядро Linux

Ядро – це свого роду головна програма, яка є основною частиною операційної системи. Воно виступає в ролі посередника між пристроями комп'ютера (процесором, відеокартою, оперативною пам'яттю і т.д.) і його програмним забезпеченням, абстрагуючись від звичайних програм і користувачів складну, низькорівневу роботу з «залізом» комп'ютера, надаючи натомість простий, зрозумілий і зручний у використанні інтерфейс. Для цього в код ядра були включені драйвери пристроїв, які можуть як завантажуватися в пам'ять разом з ядром ОС, так і підключатися в міру виникнення потреби в ресурсах необхідного пристрою.

На комп'ютері може бути запущено відразу декілька програм: якісь з них працюють у фоновому режимі, інші можуть очікувати певних дій від користувача, а третім необхідно отримувати інформацію з іншої запущеної програми. У такій ситуації саме ядро бере на себе функцію оптимального розподілу ресурсів комп'ютера між запущеними програмами і організацію паралельної роботи безлічі різних процесів. Воно першим завантажується в оперативну пам'ять комп'ютера і завжди знаходиться в занедбаному

стані, постійно взаємодіючи з його апаратним забезпеченням і встановленими програмами.

Як правило, більшість ядер діляться на три типи:

- мікроядра;
- монолітні;
- гібридні.

Мікроядро - це ядро, що складається з декількох підвантажуваних в пам'ять в міру потреби незалежних модулів, що виконуються в окремих адресних просторах. По суті, в такому варіанті виконання воно не сильно відрізняється від звичайних прикладних програм. До переваг даного ядра можна віднести теоретично більшу надійність в порівнянні з іншими архітектурою і його модульність (легкість в підключенні додаткових частин ядра). До мінусів мікроядерної архітектури відноситься те, що ядро, побудоване за такою схемою, виходить дуже повільним (адже йому потрібно постійно перемикатися між окремими частинами).

Монолітне ядро – це повна протилежність мікроядра, тому що в пам'яті комп'ютера завжди знаходиться весь (або майже весь) код ядра, внаслідок чого швидкість його роботи вище в порівнянні з мікроядром.

Гібридне ядро – це ядро, що поєднує у собі елементи як монолітної, так і микроядерной архітектур.

Ядро Linux хоч і відноситься до монолітним ядрам, але воно також запозичує і деякі ідеї з микроядерной архітектури, що означає, що вся операційна система працює в просторі ядра, а драйвера пристроїв (у вигляді модулів) можуть бути легко завантажені (або вивантажені) прямо під час роботи операційної системи [7].

2.3 Рекомендації щодо вибору дистрибутивів

Багато Дистрибутиви GNU / Linux вибирати. Це і позитивно, і негативно. З одного боку, чудово мати безліч дистрибутивів під рукою, орієнтованих на дуже конкретних користувачів, з безліччю попередньо встановлених інструментів під рукою для того, що ви зазвичай робите щодня. Спосіб задовольнити багато типів користувачів, на відміну від інших унікальних систем, таких як macOS, Windows тощо.

Але з іншого боку є вже відоме роздробленість, що ускладнює розробникам створення пакетів для всіх них. На щастя, багато дистрибутивів походять від інших, тому вони сумісні з пакетами материнського дистрибутиву, тобто є сім'ї, сумісні між собою, і не всі вони створені з нуля. Це справді була дилема ще недавно: щасливі користувачі з багатьма дистрибутивами і щасливі розробники пакетів з кількома. З надходженням універсальних пакетів ця проблема також була значно пом'якшена.

Критерії відповідно до ваших потреб:

Надійність і стійкість: Деяким людям через їхню роботу потрібна операційна система, яка є якомога стабільнішою та надійнішою. Це нестабільно і знижує продуктивність праці або змушує їх втрачати роботу, яку вони роблять через невдачі.

Безпека: Очевидно, що в наші дні ми повинні якомога ускладнити роботу кіберзлочинців. Тому операційна система також повинна бути безпечною, особливо якщо ви використовуєте її для обробки конфіденційних даних, для компаній, серверів тощо.

Сумісність та підтримка – не всі дистрибутиви включають підтримку всіх архітектур. В даний час з’являються ноутбуки з ARM, і є деякі інші проекти з PowerPC, крім деяких плат із RISC-V. Тому, якщо у вашому випадку ви хочете використовувати більш «екзотичну» архітектуру, яка не

є x86, вам слід уважно стежити за сумісністю дистрибутива, який ви збираєтесь вибрати.

Посилка: мається на увазі не сам тип пакету, а наявність програмного забезпечення для вашого дистрибутива. Незважаючи на універсальні пакети, не всі програмні проекти включають такий тип пакетів, тому іноді це може бути дещо обмежує. Зазвичай більшість відомих дистрибутивів мають велику кількість пакетів, і це не буде проблемою. Але, наприклад, DEB - явні фаворити в цьому випадку. Велич Debian та популярність Ubuntu багато в чому сприяли цьому.

Зручність використання- це залежить не стільки від самого дистрибутива, скільки від інших компонентів. Наприклад, менеджер пакетів (деякі простіші за інші), середовище робочого столу (або його відсутність) тощо. Тому він буде другорядним, оскільки ви можете встановити улюблене середовище робочого столу або менеджер переваг на ваш улюблений дистрибутив, навіть якщо він не використовував його за замовчуванням.

Інші аспекти: Інші речі, які можуть вам допомогти у виборі, це певні системи, які викликають симпатію та антипатію до певних користувачів, будь то щодо технічної, адміністративної, безпекової тощо Завжди шукайте ту, з якою вам комфортніше, оскільки хороше адміністрування певної системи може змінити ситуацію незалежно від використовуваного вами програмного забезпечення.

Ubuntu

Ubuntu є одним із дистрибутивів найбільш улюблений і вживаний. Як для початківців користувачів, які шукають щось просте з гарною підтримкою апаратного забезпечення, так і для розробників, які знаходять безліч середовищ для розробки, які працюють як шарм у цьому дистрибутиві Canonical.

Se на основі Debian, і він використовує упаковку DEB, хоча, як ви добре знаєте, сьогодні він також використовує пакети оснащення. Спадщина Debian також наділила цей розподіл гарною надійністю, стабільністю та безпекою. Крім того, якщо вам подобаються ігри, ви також побачите, що це може бути чудовим варіантом.

Debian

Debian – це один із батьківських дистрибутивів, з якого походять багато інших. Цей проект єдиний одна з найбільших і найкрутіших що можна знайти у світі вільного програмного забезпечення. Чудовий стабільний, надійний та безпечний дистрибутив, за допомогою якого можна працювати вдома або створити професійний сервер.

Цей розподіл, як і Ubuntu, може стати чудовим союзником для тих, хто шукає багатофункціональна операційна система. Зараз це може бути не найпростішим з усіх для початківців користувачів.

OpenSUSE

OpenSUSE також може служити загальним розподілом для всіх видів використання. Це стабільний, надійний та безпечний дистрибутив. Проект, який підтримується громадою, не слід плутати з його "старшою сестрою" SUSE [8].

2.4 Висновки до розділу 2

Плюси операційної системи Linux:

1) Безкоштовність

Ядро Linux і основні компоненти, з яких складається система, і безліч програм поширюються з відкритим вихідним кодом абсолютно безкоштовно. Ви можете завантажити дистрибутив Linux, наприклад Ubuntu, не заплативши за це ні рубля, і встановити його на свій комп'ютер повністю легально. У наших реаліях, де і Windows можна завантажити і

встановити безкоштовно, може здатися, що різниці немає, але варто згадати, що піратські збірки можуть бути не безпечні, та й ключ до операційної системи в будь-який момент може стати недійсним. Тут у Linux незаперечну перевагу.

2) Настроюваність

З огляду на відкритий вихідний код, при наявності певних знань ви можете змінити в системі все, що завгодно, і так, як вам захочеться. Існують навіть графічні оточення, які ви можете налаштовувати, просто створюючи конфігураційні файли на відповідному мовою програмування. Таким чином і вийшло величезна кількість дистрибутивів Linux. Люди брали базові компоненти, з'єднували їх потрібним чином, налаштовували, і виходив дистрибутив.

3) Простота встановлювання

Популярні дистрибутиви Linux дуже прості в установці. Ту ж Ubuntu можна запустити з флешки без установки і протестувати практично всі можливості операційної системи. А сама установка Ubuntu не складніше Windows, досить тільки натискати кнопку Далі. Також система встановлюється досить швидко навіть на старому обладнанні.

4) Безпека

Через низьку популярність Linux для робочих столів і архітектури системи, зловити вірус в Linux досить складно. Якщо віруси для Windows орієнтовані на поразку користувачів, і зловити їх можна де завгодно, навіть переглядаючи сайти в інтернеті, то більшість вірусів для Linux націлені на сервери і розраховані на ручне використання проти обраних цілей і конкретних програм.

Створення вірусів іншого типу для Linux зараз просто не вигідно. Та й завдяки різноманітності дистрибутивів і їх конфігурацій складно буде створити віруси, які будуть працювати скрізь. Тут відразу згадуються

історії, коли багато хто з вірусів для Windows відразу ж переставали працювати, якщо перенести систему на диск D.

5) Невибагливість до ресурсів

Linux дуже невимогливий до ресурсів. Ви можете запустити Linux без графічного оточення на сервері з дуже слабким процесором і 100 мегабайтами оперативної пам'яті, і все буде працювати. Що стосується домашнього використання Linux, то існує безліч оточень робочого столу, як вимогливих до ресурсів, так і дуже легких, з яких ви можете вибрати те, що потрібно. З особистих спостережень можу сказати, що по відчуттях Linux працює спритніше Windows на одному і тому ж двоядерному Athlon і жорсткому диску.

6) Драйвери обладнання

Ядро Linux містить всі вільні драйвери обладнання, з яким може працювати Linux. Таким чином, якщо обладнання буде працювати в Linux, то швидше за все, воно буде працювати з коробки. Також можна спробувати встановити пропрієтарні драйвери, це потрібно тільки для відеокарти і деяких принтерів і Wi-Fi-адаптерів, але далеко не всіх.

Хоча останнім часом розробники почали видаляти з ядра підтримку старих материнських плат і архітектур процесорів, все ще підтримується дуже багато старого обладнання, і все це підтримується з коробки, вам не потрібно нічого додатково встановлювати.

7) Зручна командна строка

За допомогою терміналу Linux можна зробити все і навіть набагато більше, ніж в графічному інтерфейсі. Завдяки історії команд, автодоповнення команд і шляхів до файлів, пошуку по історії, операціями збідніння команд і зручним гарячих клавішах, терміналом користуватися дуже зручно, якщо звикнути. А при необхідності можна писати цілі скрипти на Bash для автоматизації дій. В останніх версіях Windows

Microsoft теж намагається зробити термінал нормальним, але в Linux він такий вже дуже давно.

8) Зручне встановлення програм

Як такий, магазин додатків Windows з'явився тільки в Windows 8 і встановити звідти можна далеко не все, а лише кілька популярних програм. Всі інші програми необхідно встановлювати, завантажуючи виконувані файли з інтернету. У Linux більшість програм можна встановити через вбудований центр додатків або через термінал з репозиторіїв дістірбутіва.

Вам практично не знадобиться завантажувати пакунки програм в інтернеті, хіба що для установки найновіших версій і для програм, яких немає в репозиторіях, а в репозиторіях є дуже багато. А для установки того, чого немає, сущетствуют свої репозиторії, які можна підключити до системи. Також недавно з'явилися універсальні формати пакетів snap і flatpack зі своїми репозиторіями, в яких також є більшість популярних програм, яких немає в офіційних репозиторіях, наприклад, той же Viber, Telegram, Visual Studio Code, Atom і багато інших.

9) Великий вибір графічного оточення

На відміну від Windows, де є тільки провідник, в Linux є безліч графічних оточень. Це Gnome, KDE, LXDE, Enlightenment, і багато інших. Всі вони виглядають по різному, споживають різну кількість ресурсів і по різному поводяться. Кожен зможе вибрати те, що йому більше до вподоби. Також є кілька оточень, заснованих на вже існуючих, наприклад, це Cinnamon і Panteon, засновані на Gnome.

10) Продумана файлова система

В Linux немає такого поняття, як диск C і диск D. Є одна цілісна файлова система, яка починається з кореня /. Всі диски, зовнішні пристрої, віртуальні файлові системи, підключаються (монтуються) в неї. Оскільки сюди підключаються віртуальні файлові системи з настройками ядра, ви

можете взаємодіяти з ядром операційної системи, просто редагуючи файли, як зі звичайною конфігурацією.

11) Зручна система зберігання налаштувань

У Windows всі налаштування зберігаються в реєстрі. Можливо, спочатку, з точки зору продуктивності, це було гарне рішення. Однак з накопиченням в реєстрі величезної кількості записів від різних програм, це твердження стає сумнівним, а складна структура реєстру виключає можливість очищення зайвих записів. У Linux вся конфігурація програм зберігається в папці / etc /. Кожна програма створює свій файл і зберігає там настройки. Ви можете відкрити файл настройок потрібної програми і змінити значення, які вас цікавлять, а також в цьому каталозі дуже просто орієнтуватися.

12) Підтримка великої кількості архітектури

Оскільки ядро і компоненти операційних систем Linux поширюються під вільною ліцензією, вони були перенесені на безліч різних архітектур, серед яких не тільки x86 і ARM, але і такі менш відомі архітектури, як MIPS і PowerPC. І важливо відзначити, що на ARM ви отримаєте той же Linux, і ті ж програми, що і на x86-архітектурі. Наприклад, на Raspberry Pi ви отримаєте майже такий же Linux і ті ж програми.

13) Відсутність збору даних

Windows збирає статистичні дані та дані про використання комп'ютера і відправляє все це на сервера Microsoft. У Linux такої поведінки немає, була кілька років тому у Ubuntu проблема з відправкою пошукових запитів в Amazon, але вона була швидко вирішена. Також будь-який анонімний збір даних може бути просто відключений в настройках.

14) Велика кількість безкоштовних програм

Для Linux існує величезна кількість вільних і безкоштовних програм, які ви можете використовувати і які зможуть замінити більшість програм з

Windows. Причому для кожної програми, яку треба замінити, існує кілька аналогів.

Мінуси операційної системи Linux:

1) Складність засвоєння

Як би там не було, Linux дуже сильно відрізняється від Windows, тому спочатку буде складно його освоювати і звикати до нової концепції операційної системи. Це та ж файлова система, репозиторії пакетів, політика безпеки, де постійно треба вводити пароль для адміністративних дій. Деякі речі до сих пір краще робити через термінал і так далі.

2) Відсутність версій популярних програм

Це основний недолік, через який багато користувачів все ще не можуть повністю перейти на Linux. Для цієї операційної системи Microsoft не випустив свій офіс, а Adobe свій Photoshop. Також немає інших специфічних програм, таких як Компас, AutoCAD, KeyCollector та інших. Цей список можна ще продовжити. Для багатьох програм є аналоги і навіть можна намагатися запускати їх в прошарку сумісності з Windows, але це не замінить повноцінний запуск програми.

По суті, ми працюємо не з операційною системою, а з програмами, які в ній виконуються, і якби були всі необхідні програми, у Linux було б набагато більше користувачів.

3) Відсутність підтримки деяких програм

Драйвери обладнання для Linux розробляються або виробником, або ентузіастами. Протягом усього розвитку Linux багато виробників обладнання не хотіли випускати драйвери для цієї операційної системи, а ентузіасти робили свої драйвери далеко не для всього обладнання.

В основному, це були мережеві адаптери і принтери. В останні роки ситуація покращилася. Для принтерів є уніфікований стандарт, але з вибором Wi-Fi-адаптерів все ще треба бути обережним і дивитися, чи підтримуються вони операційною системою [9].

3 ПОРЯДОК ПОБУДОВИ КОМП'ЮТЕРНИХ КЛАСТЕРІВ ДЛЯ МОДЕЛЮВАННЯ НАДСЛАДНИХ ПРОЦЕСІВ НА БАЗІ ОПЕРАЦІЙНОЇ СИСТЕМИ LINUX

3.1 Вихідні дані для побудови комп'ютерного кластеру

Постановка задачі:

Задано:

– вимоги до програмного забезпечення;

– призначення та основні варіанти застосування комп'ютерного кластеру;

– перелік існуючого мережевого обладнання.

Необхідно:

– розробити етапи побудови комп'ютерного кластеру для виконання надскланих процесів на базу операційної системи Linux.

Обмеження:

– характеристики мережі доступу розраховуються на основі існуючого телекомунікаційного обладнання відомих фірм виробників та згідно вимог керівних документів;

– час на розробку проекту та розрахунки техніко-економічних показників визначає замовник.

3.2 Етапи побудови комп'ютерного кластеру

Перед виконанням будь-яких дій по конфігурації надзвичайно важливо мати хороший проект. Дані складається з двох частин:

Етап 1. Фізична схема:

- План стійки для кожного їх типу (наприклад, керуючі і обчислювальні стійки).

- Поверховий план розташування стійок під час процесу установки системи і при робочому використанні, якщо вони відрізняються.
- Схеми внутрішніх з'єднань стійок для мережі, ланцюгів харчування, пульта оператора і т.д.
- Схема зовнішніх з'єднань для серверів системи зберігання, термінальних серверів і т.д.

Логічна схема:

- Схема мережі, включаючи діапазони IP-адрес, конфігурацію підмереж, угоди по найменуванню комп'ютерів і т.д.
- CSM-конфігурація по розташуванню користувальницьких сценаріїв, апаратні настройки і вимоги з моніторингу.
- Вимоги до операційних систем, список спеціалізованих пакетів і параметри конфігурації системи.
- Схема системи зберігання даних, включаючи схему файлової системи, розбиття дисків, параметри реплікації і т.д.

Приклад кластера (див. Рис. 3.1) складається з комп'ютерів IBM Systems, що працюють на процесорах Intel® або AMD, з підключеними підсистемами TotalStorage Для простоти з'єднання в кластері виконані мідним кабелем стандарту гігабітний Ethernet. Цей кабель забезпечує хорошу швидкість в більшості випадків. Пропускну здатність між стійками можна підвищити шляхом використання bonded / port-channeled / etherchannel (вставте ваш улюблений термін для позначення методу транкинга).

Мережева топологія має форму зірки – все стійки підключені до основного комутатора керуючої стійки. У прикладі використовується три мережі: одна для управління / даних (обчислювальна мережа), одна для кластерної файлової системи (мережа зберігання даних) і одна для адміністрування пристроїв. Перші дві мережі – це звичайні IP-мережі. Для більшості завдань, включаючи взаємодії між процесами (наприклад, MPI) і

управління кластером, використовується обчислювальна мережа. Мережа зберігання даних використовується виключно для доступу і взаємодії з кластерної файлової системою.

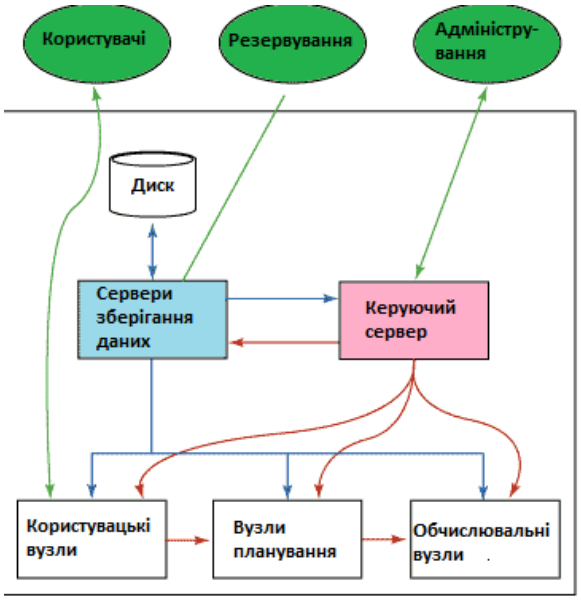

Рисунок 3.1 – Схема архітектури кластера

Додаткові подробиці схеми і дизайну прикладу кластера:

- керуючий сервер – функція керуючого сервера може виконуватися одним сервером або на кількома. У середовищі з одним сервером керуючий сервер функціонує в автономному режимі. Можна налаштувати також керуючі сервери з високою готовністю. Для цього можна використовувати програмне забезпечення CSM для підтримки високої готовності (high-availability - HA), яке видаватиме тактові імпульси ("heartbeat") між двома серверами і підтримувати динамічний відновлення після збоїв при виникненні аварійних ситуацій. Іншим можливим методом організації декількох керуючих серверів є використання реплікації, якщо підтримка HA не важлива для вашої середовища. В цьому випадку ви можете резервувати дані керуючого сервера на іншу робочу систему, яку можете, при необхідності, перевести в оперативний режим вручну. На малюнку 1 з'єднання керуючої мережі виділені червоним кольором.

Керуючий сервер – це CSM-сервер, що використовується виключно для внутрішнього управління кластером за допомогою CSM-функцій: управління установкою системи, моніторинг, облуживание і інші завдання. В даному кластері присутня тільки один керуючий сервер;

- сервери зберігання даних і дискові накопичувачі – ви можете підключити декілька серверів зберігання даних до організованого на дискових накопичувачах сховища даних за допомогою різних механізмів. Підключити систему зберігання даних до сервера можна безпосередньо: або через SAN-комутатор (storage area network - мережа зберігання даних) через оптичне волокно або мідному кабелю, або використовуючи обидва типи з'єднань. Ці сервери надають спільний доступ до системи зберігання даних інших серверів кластера. Якщо необхідно резервування бази даних, підключіть резервне пристрій до сервера зберігання даних, використовуючи додаткове мідне або оптичне з'єднання. У прикладі кластера сховище являє собою єдину сутність, що забезпечує доступ до загальної файлової системи в межах кластера. У наступній статті даної серії буде більш детально описана установка, настройка і реалізація апаратного забезпечення системи зберігання даних і кластерної файлової системи;
- призначені для користувача вузли – в ідеальному випадку обчислювальні вузли кластера не повинні приймати зовнішні підключення і повинні бути доступні тільки для системних адміністраторів через керуючий сервер. Користувачі системи можуть зареєструватися на обчислювальних вузлах (або вузлах реєстрації) для виконання своєї роботи в кластері. Кожен призначений для користувача вузол складається з образу з можливостями будь-якого редагування, необхідних бібліотек розробника, компіляторів і всього, що необхідно для створення кластерного додатки і отримання результатів;

- вузли планування – для запуску робочого навантаження на кластері користувачі повинні передати свою роботу вузла планування. Фоновий процес-планувальник (scheduler daemon), що працює на одному або декількох вузлах планування, застосовує визначену політику для запуску робочих навантажень в кластері. Аналогічно обчислювальним вузлам, вузли планування не повинні приймати зовнішніх підключень від користувачів. Системні адміністратори повинні управляти ними за допомогою керуючого сервера;
- обчислювальні вузли – ці вузли виконують робоче навантаження кластера, приймаючи завдання від планувальника. Обчислювальні вузли – це самі вільні частини кластера. Системний адміністратор може легко встановлювати заново або перенастроювати їх за допомогою керуючого сервера;
- зовнішні з'єднання – приклад зовнішнього з'єднання показаний зеленим кольором на рисунку 3.1.

Такі сполуки вважаються зовнішніми для кластера, а тому не описуються в даній статті.

Етап 2. Апаратна конфігурація

Після збору стійки і приміщення її на своє місце з усіма підключеними кабелями все одно залишається досить роботи з налаштування апаратного забезпечення. Спеціалізовані подробиці кабельних з'єднань будь-якого конкретного кластера в даній статті не описуються. Дії по конфігурації апаратного забезпечення, які необхідно виконати до установки кластера, описуються на деяких специфічних прикладах, використовуючи наведену вище схему кластера.

Логічна мережева схема

Одним із завдань, зазвичай пропускаються при установці кластера, є проектування логіки мережі. В ідеальному випадку логічна схема повинна бути виконана в паперовому варіанті до введення кластера в експлуатацію.

Як тільки у вас з'явиться логічна схема, використовуйте її для створення файлу hosts. Для невеликого кластера ви можете написати файл hosts вручну. Однак зазвичай краще визначитися з угодами по найменуванню і написати спеціалізований сценарій для формування цього файлу.

Переконайтеся в тому, що всі пристрої в мережі присутні у файлі hosts. У деяких прикладах реалізації використовуються наступні компоненти (з прикладами назв):

- керуючі сервери (mgmt001 – mgmtXXX);
- сервери зберігання даних (stor001 – storXXX);
- обчислювальні вузли (node001 – nodeXXX);
- вузли планування (schd001 – schdXXX);
- призначені для користувача вузли (user001 – userXXX).

Ці угоди по найменуванню охоплюють тільки п'ять типів комп'ютерних систем в мережі і тільки для однієї мережі, що не досить добре. Існують також мережі зберігання даних і обчислювальні мережі, які потрібно брати до уваги, плюс мережа управління пристроями. Тому цей файл повинен бути розширений. Кожен вузол, якому необхідний доступ до кластерної файлової системи, повинен мати адресу в мережі зберігання даних. Кожен вузол повинен мати дві адреси в обчислювальної мережі: один для комп'ютера, а другий для контролера Baseboard Management Controller (BMC), який використовується для моніторингу апаратних ресурсів і управління живленням. У таблиці 2.1 наведені набагато більш універсальні угоди по найменуванню з прикладом діапазонів IP-адрес.

Таблиця 2.1 Угоди по найменуванню для файлу hosts

Пристрій	Обчислення 192.168.0.0/24	BMC 192.168.0.0/24	Система зберігання 192.168.1.0/24	Пристрої 192.168.2.0/24	Зовнішня мережа ext n/w
Керуючий сервер	mgmt001	mgmt001_d	mgmt001_s	mgmt001_m	mgmt001_e
Сервер зберігання даних	stor001	stor001_d	stor001_s	stor001_m	stor001_e
Користувацькі вузли	user001	user001_d	user001_s	ні	ні
Вузли планування	schd001	schd001_d	schd001_s	ні	ні
Обчислювальні вузли	node001	node001_d	node001_s	ні	ні
Комутатори обчислювальної мережі	ні	ні	ні	gogb001a	ні
Комутатори мережі зберігання даних	ні	ні	ні	gogb001b	ні
Термінальні сервери	ні	ні	ні	term001	ні
Контролер пристроїв зберігання A/B	ні	ні	ні	disk01a/b	ні
LSM/KVM/RCM	ні	ні	ні	cons001	ні

При реалізації цієї схеми формується файл hosts, аналогічний наприклад, посилання на завантаження якого ви можете знайти в розділі "Завантаження". Це приклад невеликого кластера, що складається з шістнадцяти обчислювальних вузлів, одного керуючого сервера, одного сервера зберігання даних, одного призначеного для користувача вузла і одного вузла планування, розташованих в двох стійках разом з усіма необхідними підключеними пристроями. Ми не наводимо приклад великого кластера, оскільки такій конфігурації досить для нашого кластера, і ви легко можете розширити його при необхідності.

Ethernet-комутатори

Є дві фізичних мережі: одна для обчислень, а друга для зберігання даних. Стандартна ємність стійки в 32 вузла вимагає застосування двох 48-

портових комутаторів на кожну стійку, по одному на кожну мережу. У маленьких кластерах в керуючої стійці також необхідно використовувати два однакових комутатора. Для великих кластерів 48-портів може виявитися недостатньо, і може знадобитися більш потужний центральний комутатор.

Кожен комутатор для двох основних мереж (проігноруймо мережу управління пристроями) вимагає трохи різного конфігурації, оскільки, як в даному прикладі, Gigabit Ethernet для взаємодії використовує джамбо-пакети (jumbo frames) в мережі зберігання даних і пакети стандартного розміру для обчислювальної мережі. Налаштування мережі управління пристроями зазвичай є дуже простий: одноуровневого дворежимного мережевого комутатора 10/100 досить для керування пристроями, тому більш докладне пояснення зайве.

Етап 3. Приклад :Комутатор Extreme Networks

Нижче перераховані дії по конфігурації 48-портового Gigabit Ethernet комутатора Extreme Networks Summit 400-48t.

Перш за все, підключіться до кожного комутатора, використовуючи послідовний порт і стандартний послідовний кабель (9600, 8-N-1, без управління потоком); ID користувача за замовчуванням - admin, пароль відсутній (просто натисніть клавішу Enter при запиті).

Для всіх комутаторів виконайте наступні дії:

1. Вводимо unconfig switch all – очистити будь-яку наявну настройку, при необхідності.

2. Вводимо configure vlan mgmt ipaddress 192.168.2.XXX/24 – встановити IP-адреса для управління.

3. Вводимо configure snmp sysname gigbXXX.cluster.com – встановити ім'я комутатора.

4. Вводимо configure sntp-client primary server 192.168.2.XXX – встановити NTP-сервер для керуючого сервера.

5. Вводимо configure sntp-client update-interval 3600 – встановити погодинне час синхронізації.

6. Вводимо configure timezone 0 – встановити тимчасову зону.

7. Вводимо enable sntp-client – включити NTP.

8. Вводимо configure ports 1-4 preferred-medium copper – змінити бажаний матеріал сполучних кабелів за замовчуванням з оптоволокна на мідний дріт для портів 1-4, при необхідності.

Тепер налаштуємо джамбо-фрейми на комутаторі мережі зберігання даних, виконавши такі дії:

9. Вводимо create vlan jumbo – створити джамбо-фрейми vlan.

10. Вводимо configure "mgmt" delete ports 1-48 – видалити порти з mgmt vlan.

11. Вводимо configure "jumbo" add ports 1-48 – додати порти в jumbo vlan.

12. Вводимо configure jumbo-frame size 9216 – встановити максимальний розмір пакета передачі (MTU - maximum transmission unit).

13. Вводимо enable jumbo-frame ports 1-48 – дозволити підтримку джамбо-фреймів.

Щоб дозволити транкінг по двох портів, використовуємо команду enable sharing 47 grouping 47-48 (групує разом порти 47 і 48; порт 47 використовується в якості первинного).

Для компіляції конфігурації виконайте наступні дії:

14. Вводимо save configuration primary – записує конфігурацію комутатора у флеш-пам'ять для використання після перезавантажень.

15. Вводимо use configuration primary [10].

Для приклада роботи кластера продемонструємо на прикладі кластера "Уран" (Рис. 3.2) з використанням програм WinSCP і PuTTY

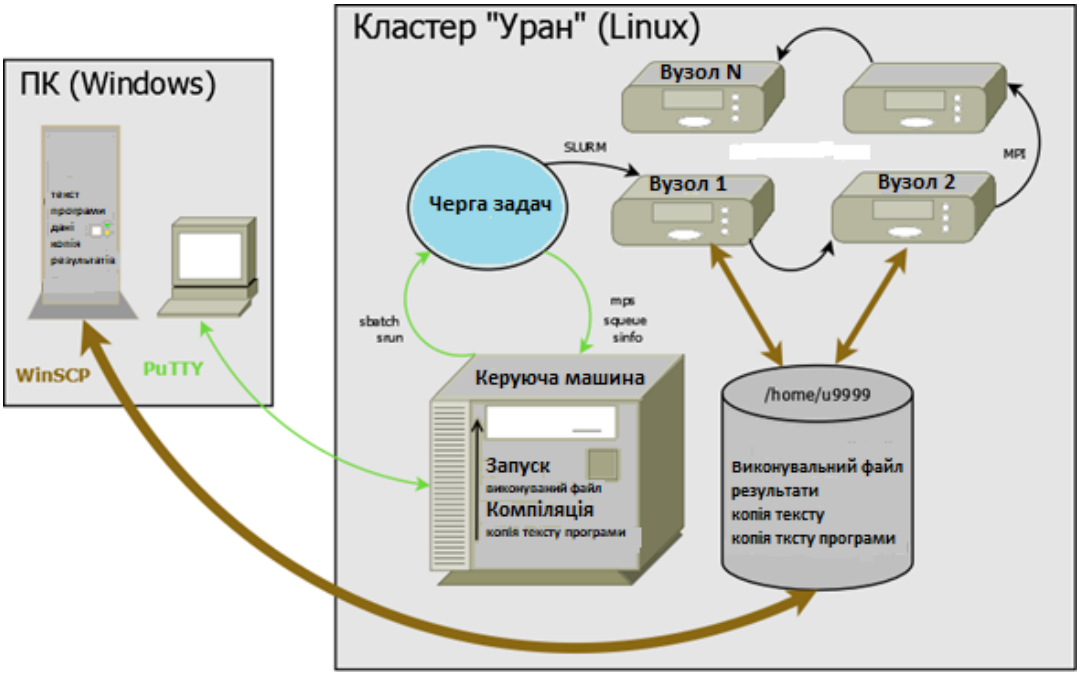

Рисунок 3.2 – Кластер «Уран»

Для обчислень на комп'ютері або кластері потрібен виконуваний файл програми і, як правило, вихідні дані. Оскільки на комп'ютері користувача зазвичай встановлена операційна система Windows, а на кластері - Linux (різновид UNIX), то для рахунку на кластері необхідно отримати виконуваний файл за допомогою одного з компіляторів кластера. Отже, на кластер повинен бути переписаний текст програми.

Для віддаленого копіювання файлів використовується програма WinSCP.

Користувач копіює файли (текст програми, вихідні дані) з персонального комп'ютера в свій домашній каталог на кластері:

на рисунку 3.2 це копіювання в / home / u9999, де u9999 – ім'я (ідентифікатор, login) користувача на кластері.

Компіляція програм і запуск завдань на рахунок здійснюються після входу на кластер через програму віддаленого доступу PuTTY. Набираючи в вікні PuTTY відповідні команди, користувач виконує на керуючу машину (хост) потрібні йому дії, зокрема, компіляцію і запуск завдань на кластері. При цьому запуск завдання здійснюється за допомогою постановки її в

чергу на рахунок. Веденням черзі і стартом завдань на обчислювальних вузлах займається система SLURM. Під час рахунку паралельні обчислювальні процеси завдання можуть обмінюватися даними по комунікаційної мережі і мають доступ до домашнього каталогу.

Користувач в будь-який момент може отримати інформацію про свої завдання в черзі і про доступні ресурси кластера.

Прискорення обчислень досягається лише після перетворення послідовної програми в паралельну, зазвичай із застосуванням стандартів MPI і OpenMP. Запуск задач на кластері здійснюється з урахуванням особливостей програм, зокрема, з урахуванням використання MPI і / або OpenMP. Для налагодження корисно запустити на кластері послідовний тестовий варіант програми. Так само як паралельні, послідовні (однопроцессной) програми повинні запускатися шляхом постановки в чергу.

Для роботи на кластері корисно ознайомитися з базовими командами ОС UNIX.

Вхід на кластер по секретному ключу:

-доступ по ключу;

- використання агента аутентифікації;

- генерація ключів в UNIX;

- спільне використання ключа і пароля.

Для входу на кластер користувач повинен мати логін (ім'я користувача) і для аутентифікації (підтвердження автентичності) ключ або пароль (password).

Оскільки використання ключа забезпечує більшу конфіденційність і дає можливість позбутися від підтвердження пароля для операцій типу scp (віддаленого копіювання файлів), доступ по ключу розглядається в подальшому як більш кращий.

Тому спочатку, а також при втраті пароля або ключа для забезпечення доступу до кластеру необхідно створити ключі, як це описано нижче.

Для входу на кластер з ОС Windows можна скористатися програмою PuTTY.

Етап 4. Доступ по ключу

Для організації доступу по ключу потрібно виконати наступну послідовність дій:

1. Згенерувати і зберегти на своєму комп'ютері приватний (секретний) і публічний ключі, використовуючи програму PuTTYgen з поставки PuTTY, як проілюстровано в інструкції [1]. При цьому публічний ключ слід зберегти в файлі Блокнота, копіюючи текст з PuTTYgen: починаючи з типу ключа 'ssh-rsa' і закінчуючи знаками '==' і коментарем з датою виду 'rsa-key-20150421'.

Секретний ключ рекомендується зашифрувати кодовою фразою (паролем секретного ключа, passphrase) і зберігати в безпечному місці.

2. Додати публічний ключ в файл ~ / .ssh.authorized_keys каталогу на сервері ('~' позначає домашній каталог користувача). Користувач може зробити це самостійно (якщо у нього є доступ на кластер по паролю або іншому ключу) або відправити файл з публічним ключем адміністратору кластера на адресу Адміністратор кластера встановить публічний ключ і повідомить про це користувачеві.

3. Створити в PuTTY іменовану сесію (Session) для роботи на кластері з доступом по ключу, для чого вставити шлях до секретного ключа в розділі Connection-> SSH-> Auth, повернутися в розділ Session і, заповнивши поля Host Name і Saved Sessions значеннями. Після цього робота з кластером може бути ініційована подвійним натисканням на імені створеної сесії.

Використання агента аутентифікації.

Якщо секретний ключ захищений кодовою фразою (passphrase), то запуск агента аутентифікації, програми Pageant з поставки PuTTY, позбавляє від введення passphrase при повторному з'єднанні по ключу з кластером на період часу до завершення роботи агента.

Pageant можна запустити, наприклад, подвійним кліком на секретному ключі і набором його кодової фрази (якщо є). В результаті в системній області панелі завдань (внизу праворуч) з'явиться іконка Pageant. Далі (при кожному вході на кластер) правою кнопкою миші відкриваємо меню і вибираємо для запуску потрібну сесію (umt).

Етап 5. Налаштування аутентифікації по ключу в PuTTY

Технологію доступу по ключу можна алегорично представити як якби секретний ключ був таким маленьким красивим фігурним ключиком з вигадливою борозенкою, а публічний – це замочок зі свердловиною точно під цей ключ, який потрібно встановити на віддалений сервер.

Відкрийте програму Puttygen і натисніть кнопку Generate. Для процесу генерації ключів програма вимагає здійснювати довільні руху "мишею" – для заповнення масиву випадкових чисел (Рис. 3.3).

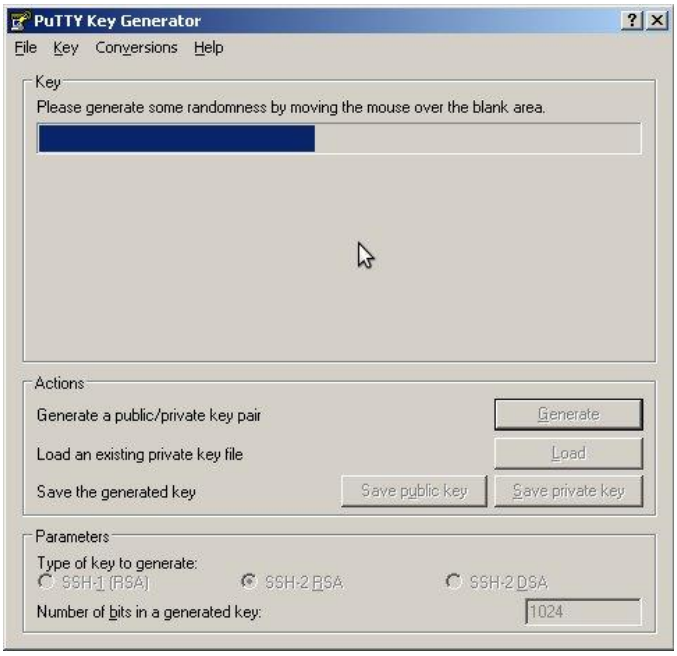

Рисунок 3.3 – Вікно для заповнення масиву випадкових чисел

Збережемо створений секретний ключ Save secret key. При збереженні слід врахувати, що за допомогою цього ключа можна отримати доступ до всіх ресурсів, доступ до яких Ви налаштуєте. Спробуйте не зберігати його на комп'ютері, куди хоча б іноді може мати доступ хтось ще. Збережіть наприклад його на флешці. Буде правильним, якщо ключ буде захищений паролем (Рис. 3.4)

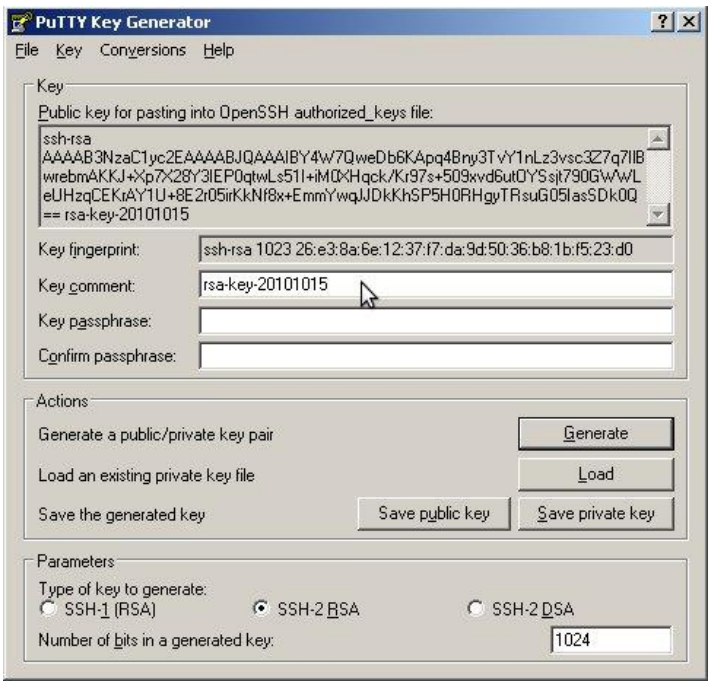

Рисунок 3.4 – Вікно введення захищеного паролю

Виділяємо мишею відкритий ключ OpenSSH як показано на рисунку і скопіюйте його в буфер обміну (Рис. 3.5)

Відкрийте PuTTY і встановіть з'єднання з сервером використовуючи пароль. Створіть на сервері каталог .ssh командою

mkdir .ssh

Потім виконайте команду echo як показано на рисунку - наберіть

echo '

За замовчуванням, якщо клікнути після цього правою кнопкою миші по консолі, в ній з'явиться вміст буфера обміну - тобто збережений раніше

відкритий ключ. Далі закриваємо лапки і вказуємо перенаправлення в файл:

'>> .ssh / authorized_keys

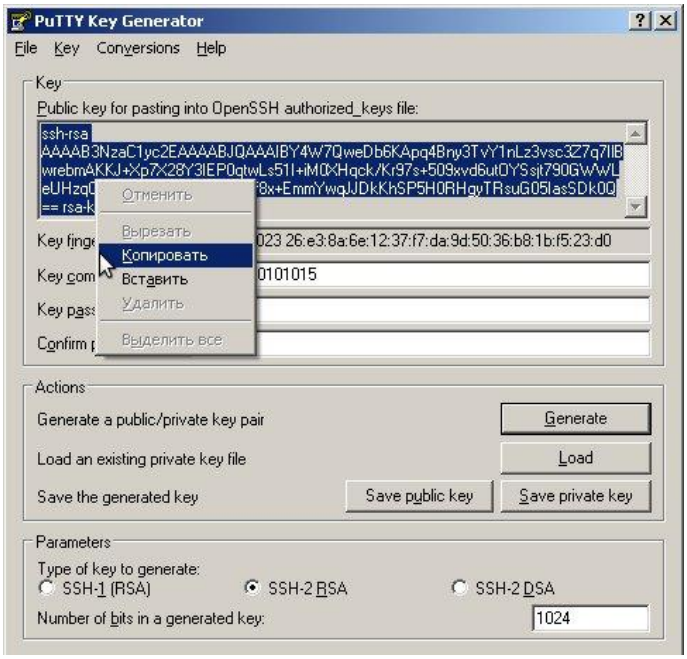

Рисунок 3.5 – Вікно процесу копіювання ключа

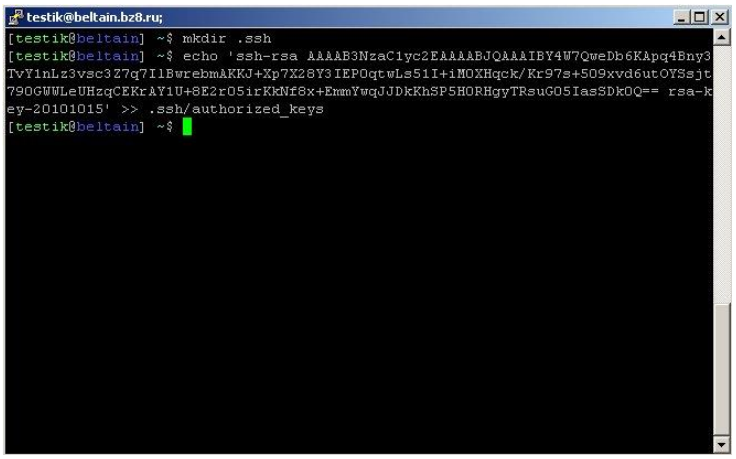

Рисунок 3.6 – Вікно командного рядка

Налаштуйте PuTTY для роботи з секретним ключем. Для цього увійдіть в розділ Connection-> SSH-> Auth і виберіть шлях до збереженому ключу (Рис. 3.7).

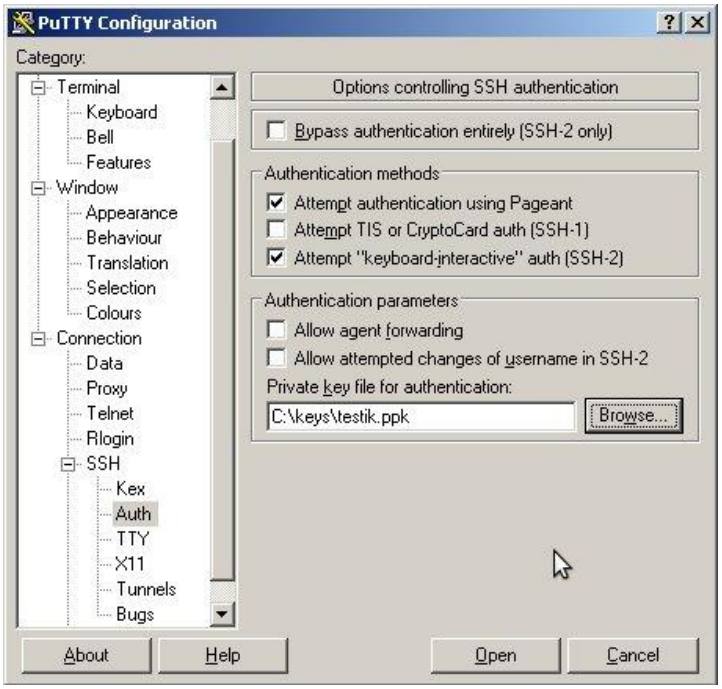

Рисунок 3.7 –Вікно налаштування конфігурації

Не забуваємо зберегти налаштування(Рис. 3.8).

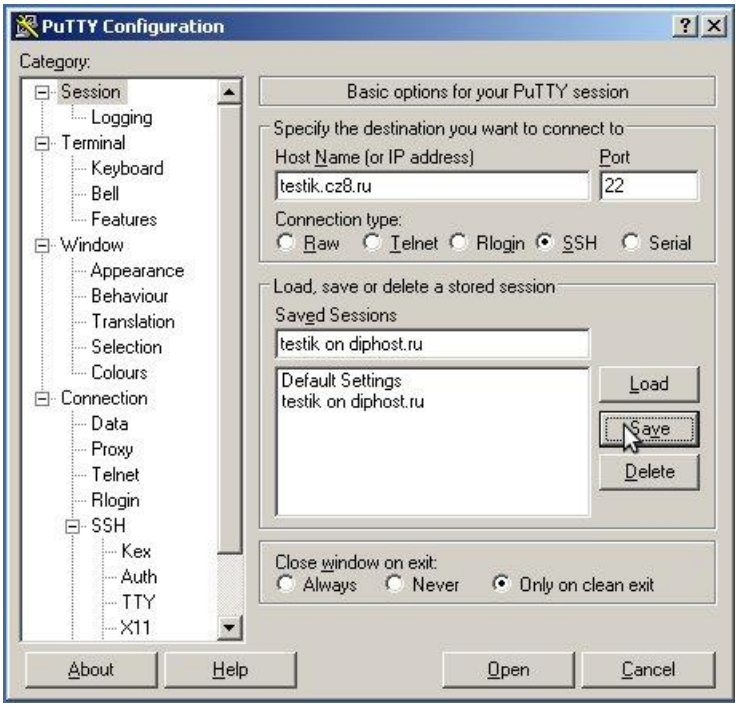

Рисунок 3.8 – Вікно збереження конфігурацій

Перевіряємо, якщо все налаштоване правильно, PuTTY більше не буде питати пароль при даному підключенні, не рахуючи запиту на пароль до самого ключу.

Етап 6. Генерація ключів в UNIX

В UNIX-системах для генерації ключів використовується утиліта ssh-keygen, створює два файли: ~ / .ssh / id_rsa і ~ / .ssh / id_rsa.pub (імена за замовчуванням), агент аутентифікації запускається за допомогою утиліти ssh-agent.

Спільне використання ключа і пароля

Маючи доступ по ключу, користувач при необхідності може запросити пароль у адміністратора кластера. Адміністратор створює пароль (password) і кладе його в домашній каталог користувача, повідомляючи йому про це.

Користувач завжди може змінити пароль командою passwd.

Треба мати на увазі, що при вході на кластер спочатку робиться спроба пройти аутентифікацію по ключу, а в разі невдачі – по паролю.

У PuTTY зручніше мати різні сесії для доступу по ключу або паролю.

При доступі по паролю вводити password потрібно завжди. У разі використання ключа кодова фраза (passphrase) запитується (якщо є), але можливо тільки один раз і можливо для доступу до багатьох систем. Тому кодова фраза може бути більш складною, що підвищує безпеку, не надто ускладнюючи роботу користувача.

Програми віддаленого доступу

Для роботи на суперЕОМ з персонального комп'ютера користувачеві необхідно встановити програми на комп'ютері, які забезпечатьвіддалений доступ до командного рядка сервера для компіляції програм і запуску завдань, можливість обміну файлами між комп'ютером і сервером і дозволять використовувати зручний графічний інтерфейс.

Етап 7. Віддалений доступ до командного рядка сервера

Доступ до серверів здійснюється через протокол SSH (власне SSH - «secure shell»). Для роботи з командним рядком рекомендується програма PuTTY. Ця програма дозволяє раз ввести адресу сервера і, в подальшому, вибирати його зі списку сесій. В налаштуваннях PuTTY необхідно вказати протокол SSH. При першому з'єднанні з сервером програма видає попередження про те, що ключ шифрування сервера раніше не використовувався і пропонує його зберегти або відкинути (продовжити сеанс або перервати). Природно треба вибрати продовжити розмову.

Корисно більш детально ознайомитися з установкою та деякими настройками програми PuTTY.

Приклад діалогу при вході на сервер UM16 з ім'ям xxxxx з комп'ютера ada.imm.uran.com:

```
login as: xxxxx
Password:                          (ввести пароль)
Last login: Tue Sep 6 16:31:23 2005 from ada.imm.uran.com
```

Ви можете змінити свій пароль в будь-який час, набравши команду passwd, наприклад:

```
[~@um16]:passwd
Password for xxxxx:                (наберіть тут свій поточний
пароль)
Enter new password:                          (наберіть новий пароль)
Enter it again:                              (повторіть новий пароль)
Password changed.                            (Успішна зміна пароля)
```

Для того щоб захистити свій обліковий запис, дотримуйтеся наступних рекомендацій:

- використовуйте пароль не менше, ніж з 6 букв і цифр і зберігайте його в секреті;
- один раз на півроку міняйте свій пароль;

- ніколи не залишайте активний термінал без уваги, завжди виходьте (logout) зі свого терміналу (закінчуйте сеанс) перш, ніж покинути його;
- обов'язково повідомляйте про будь-якому неправильному використанні або зловживанні системного адміністратора, інакше доступ до сервера буде закритий.

Для того, щоб завершити сеанс роботи на сервері, виконайте команду logout.

Обмін файлами між комп'ютером і сервером

Для обміну файлами між комп'ютером і сервером рекомендується використовувати програму WinSCP.

При копіюванні файлів (протоколи SCP - "secure copy" і SFTP - "secure ftp") в настройках WinSCP рекомендується вибрати протокол SFTP. Програма WinSCP дозволяє зберегти ім'я користувача та пароль. При першому з'єднанні з сервером програма видає попередження про те, що ключ шифрування сервера раніше не використовувався і пропонує його зберегти або відкинути (продовжити сеанс або перервати). Природно, треба вибрати продовжити розмову. Програма WinSCP може використовуватися як самостійний додаток і як плагін для інших програм провідників, наприклад для Far Commander.

Тексти програм і дані для завдання можна записати в окремий каталог на сервері, тоді вся інформація по цьому завданню буде зберігатися в цьому каталозі (об'єктні модулі після трансляції текстів програм, файл з повідомленнями про помилки, вихідні файли).

На сервері існує понятіе домашнього каталогу, це каталог з ім'ям / home / користувач. При роботі з командним рядком цей каталог стає поточним після встановлення термінального з'єднання. Програма WinSCP може запам'ятовувати останній відвіданий в попередньої сесії каталог. Тому важливо стежити, в якій каталог проводиться копіювання.

Після копіювання необхідно перейти в робочий каталог, виконати компіляцію програми і її запуск.

Корисно ознайомитися з інформацією про встановлення і деяких налаштуваннях програми WinSCP.

Етап 8. Програмне забезпечення кластерів

Програмне забезпечення кластера (транслятори, бібліотеки, пакети прикладних програм і т.д.) Оновлюється і поповнюється при виході нових версій і за запитами користувачів. Для компіляції програм доступні Intel компілятори (безкоштовно для некомерційного використання в версіях Linux), вільно поширювані компілятори серії GCC (GNU Compiler Collection) і компілятори Portland Group (PGI) з мов C, C ++, Fortran:

```
Intel    (icc, icpc, ifort)

GNU      (gcc, g++, gfortran, g77)

PGI      (pgcc, pgcpp, pgf77, pgf90, pgf95)
```

Для трансляції Паскаль-програм можна скористатися Free Pascal Compiler (fpc).

Для організації взаємодії між процесами можна використовувати такі бібліотеки обміну повідомленнями, що реалізують стандарт MPI:

```
MVAPICH2

OpenMPI

MPICH2
```

MPI (Message Passing Interface) орієнтований, перш за все, на системи з розподіленою пам'яттю. Можливо розпаралелювання програм і за допомогою стандарту OpenMP (Open Multi-Processing), орієнтованого на системи із загальною пам'яттю (див. Компіляція і запуск завдань з OpenMP), а також стандартів CUDA (див. Використання CUDA) і OpenACC (Open Accelerators), націлених на використання графічних процесорів (GPU). У компіляторах Portland Group реалізовані як OpenMP, так і OpenACC (див. PGI Accelerator і OpenACC).

На кластері встановлено програмне забезпечення (ПО) фірми Intel для профілювання і налагодження - Intel Parallel Studio XE. Це ПО дозволяє знаходити найбільш навантажені місця в додатку, підраховувати ступінь паралельності програми, знаходити тупики та гонки в паралельних програмах і т.д. Основні інструменти Advisor XE, Inspector XE і Vtune Amplifier XE знаходяться в відповідних папках в каталозі / opt / intel (див. Інструменти Intel для профілювання і налагодження).

Для зміни компіляторів і бібліотеки MPI, а також вибору інших пакетів прикладних програм використовуйте модулі установки змінних оточення [11].

Як приклад побудови відмовостійкої системи розглянемо таку задачу. Для великого веб-магазину необхідно створити портал з гарантованою доступністю 24x7 і перейти на нову архітектуру практично без зупинки працюють сервісів.

Етап 9. Приклад побудови відмовостійкої системи

1) Створення балансування на рівні мережі

Для балансування використовувався сервіс HAProxy на серверах балансування, що знаходяться в різних ЦОД, а над ними - сервіс CloudFlare з балансуванням по DNS і перевіркою по порту. Зв'язок між серверами балансування і іншими сервісами клієнта забезпечувалася одним розтягнутим L2-доменом. (Рис 3.9)

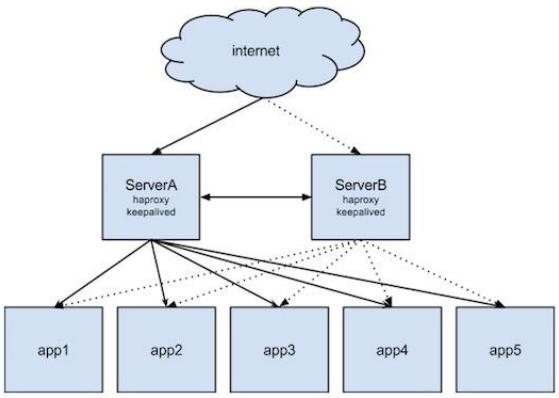

Рисунок 3.9 – Структура мережі для балансування серверів

На даному етапі робота сервісу не буде залежати від збоїв на мережевому рівні на майданчиках провайдера. У разі, якщо сервери балансування перестають бачити один одного на рівні L2-домену, але продовжують бачити один одного через мережу Інтернет – вважаємо, що стався обрив L2-зв'язності майданчиків. У цьому випадку одна з майданчиків ставати майстром (попередньо встановлена конфігурація), а друга – підлеглою. CloudFlare в цей момент вимикає другий майданчик з переліку.

Для кожного сервера балансування знадобитися 2 ГБ дискового простору, 256 МБ оперативної пам'яті і 4 процесора (будуть використовуватися тільки в піках навантаження).

2) Створення резервної балансування веб-серверів

Для резервування веб-сервера використовувався сервер синхронізації lsyncd в режимі source-destination. Для включення даного режиму потрібно буде по черзі перезапустити кожен веб-сервер з кластера.

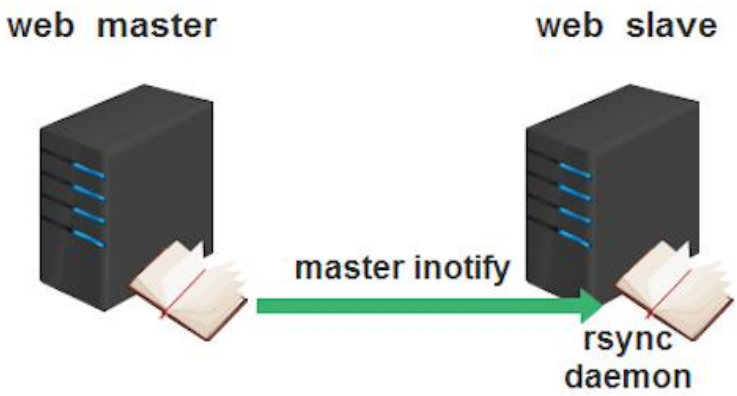

Рисунок 3.2.10 – Схема синхронізації веб серверів

При резервуванні на рівні веб-сервера навантаження може бути розподілена між серверами. Відповідно для побудови даної схеми знадобитися додатково 100% дискового простору, ресурси оперативної пам'яті і процесора розподіляються між майданчиками порівну в розмірі 60-70% від існуючої.

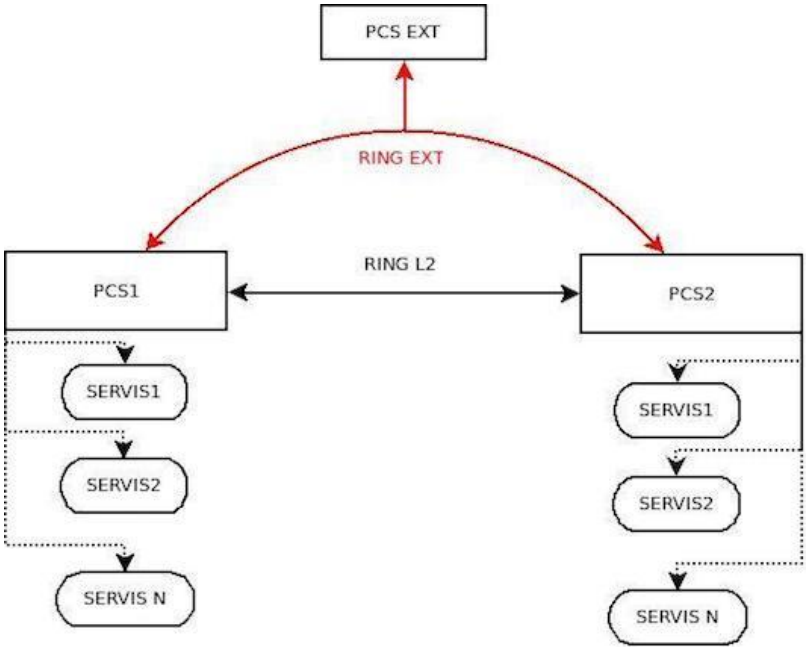

Рисунок 3.11 – Загальна схема кластера

3) Резервування і балансування БД

З отриманих даних співвідношення операцій запису і читання складають 20 до 100 при загальному рівні 75 операцій записи-оновлень в секунду.

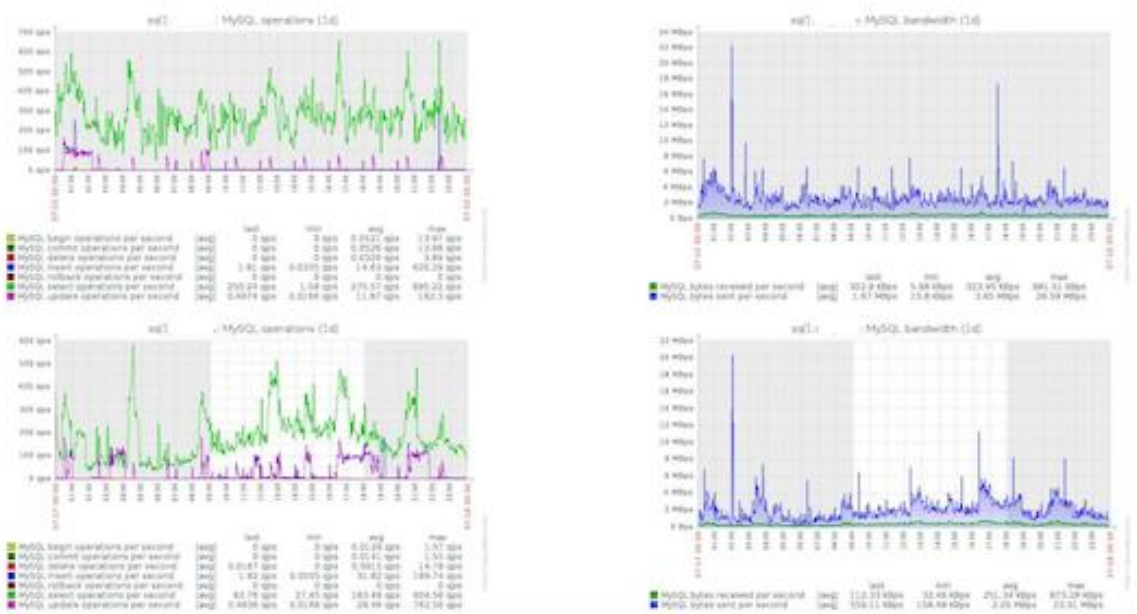

Рисунок 3.12 – Вхідна інформація про БД

За умови, що затримка мережі між двома серверами бази даних буде становити менше 1 мс, можна буде використовувати архітектуру master-master (сервери зможуть гарантовано обробити більше 1000 запитів «запис + оновлення» в секунду).

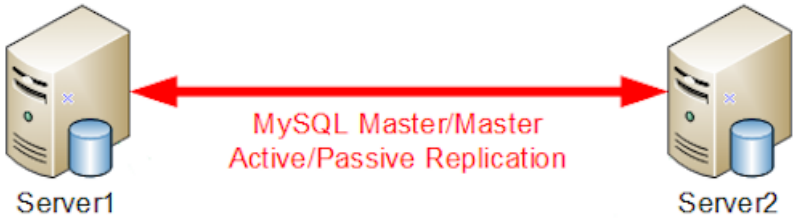

Рисунок 3.13 – Схема архітектури мастер-мастер

У разі архітектури master-slave не може бути гарантована як безперервність доступу (так як при недоступності основного сервера потрібен час на ручну або автоматичну операцію призначення нового основного сервера), так і повна схоронність даних (в разі краху основного сервера все не реплікованих дані на ведені сервера будуть втрачені).

Для архітектури master-master існують готові рішення. Це можна реалізувати над серверами Percona XtraDB або MariaDB Galera. У обох варіантів рішення є свої плюси і мінуси.

Percona XtraDB - це збірка MySQL з включеним за замовчуванням XtraDB storage engine. Відрізняється від MySQL + InnoDB plugin кращою продуктивністю / масштабованість, особливо на сучасних багатоядерних серверах. Збирається в варіантах, які базуються на MySQL 5.0 і 5.1. Повністю сумісна з таблицями innodb, тобто без проблем підтримується конвертація innodb – xtradb і назад (якщо не використовувати деякі специфічні для xtradb функції типу меншого розміру сторінки).

Приріст продуктивності помітний тільки при рівні запитів понад 1000 на запис. Але дана конфігурація менш стабільно працює в разі збоїв в роботі вузлів або втрати зв'язку між вузлами.

Для реалізації знадобитися додатково 100% дискового простору (під базу), ресурси оперативної пам'яті і процесора розподіляються між майданчиками порівну в розмірі 60-70% від існуючої.

Перевагою сервісу MariaDB Galera є стабільність і надійність роботи, підвищена стійкість і збереження цілісності даних після краху при повній сумісності з MyISAM.

За іншим характеристикам дані сервіси є аналогічними.

Було запропоновано побудувати рішення, використовуючи MariaDB Galera, так як при даному рівні запитів на запис і зміни різниці в швидкості роботи бази даних не буде, але можна буде отримати більш надійну систему. Запас масштабованості є, оскільки для досягнення тисяча запитів в секунду можна збільшити навантаження в 130 (!) Разів.

Балансування трафіку, що надходить на сервери БД

Для балансування пропонується використовувати універсальний балансувальник трафіку HAPRoxy з додатковою налаштуванням для перевірки стану DB. Цей сервіс встановлюється і запускається на веб-серверах, а не на окремих віртуальних машинах.

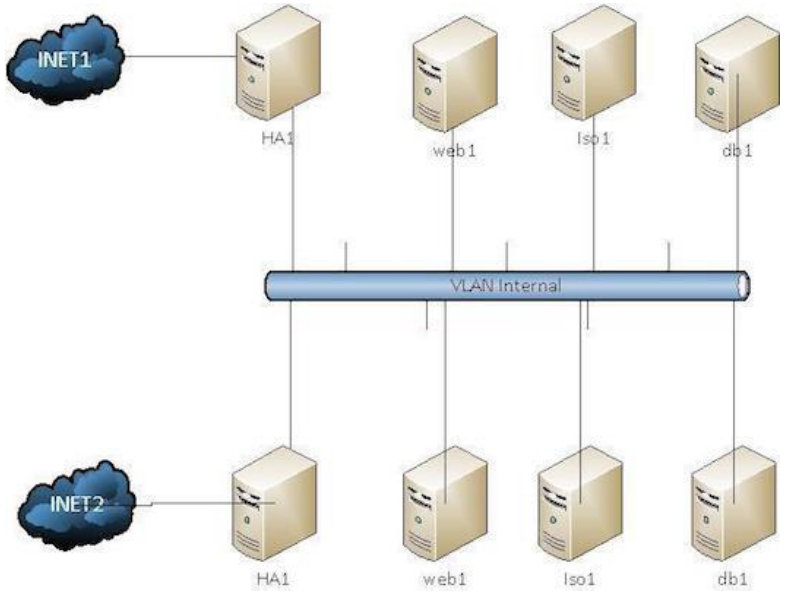

Рисунок 3.14 – Схема підключення віртуальних машин

Даний балансувальник перевірятиме доступність кожного вузла за допомогою тестових запитів і в разі невдачі виключати помилковий вузол зі списку до моменту відновлення коректної роботи. У разі рассинхронизации вузлів балансувальник перемкне режим роботи тільки з одним вузлом з максимальною версією реплікації. Балансувальник також відповідатиме за рівномірні розподілу запитів між усіма вузлами.

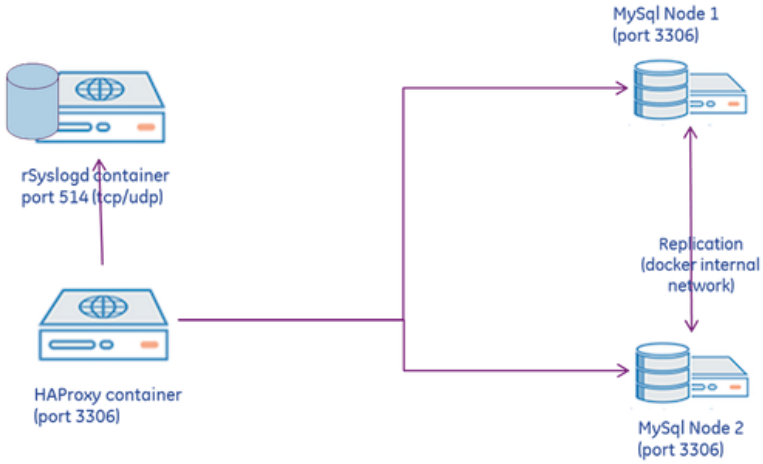

Рисунок 3.15 – Схема балансування серверів

Для максимального збереження інформації резервування можна робити на кожному вузлі незалежно.

При даній архітектурі вихід будь-якого вузла не викличе переривання в роботі сервісу, а після відновлення роботи збійного вузла потрібно буде просто виконати реплікацію даних з працюючого вузла вручну.

Для переходу на кластерну систему знадобитися короткочасна перерва в роботі сервісу для остаточної синхронізації бази даних [10].

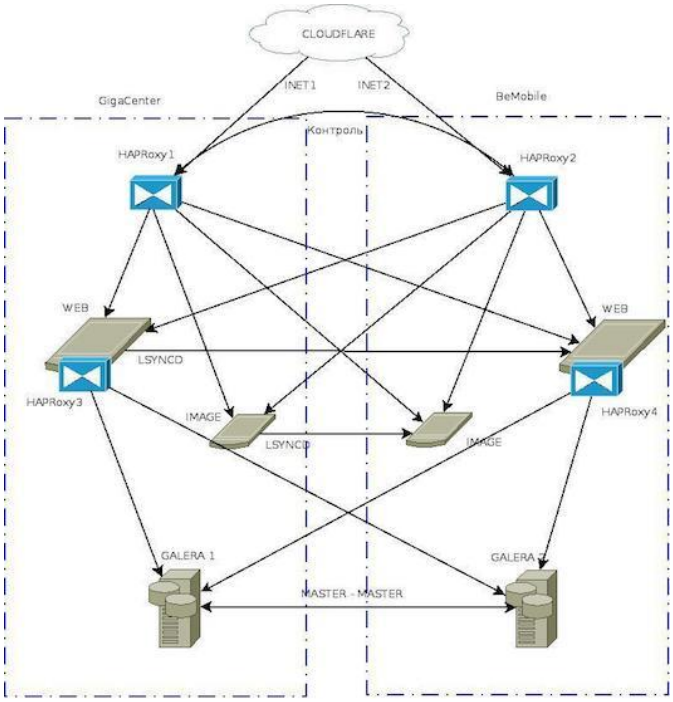

Рисунок 3.2.16 – Загальна схема взаємодії модулів

3.3 Висновки до розділу 3

В розділі розглянуто порядок формування вхідних даних та основні етапи побудови комп’ютерних кластерів на базі операційної системи Linux. Які дають змогу на базі комп’ютерної техніки створити комп’ютерний кластер для застосування в закладах вищої освіти.

Кластери використовуються в обчислювальних цілях, зокрема в наукових дослідженнях. Для обчислювальних кластерів істотними показниками є висока продуктивність процесора в операціях над числами з плаваючою точкою (flops) і низька латентність об'єднує мережі, і менш істотними - швидкість операцій введення-виведення, яка більшою мірою важлива для баз даних і web-сервісів. Обчислювальні кластери дозволяють зменшити час розрахунків, у порівнянні з одиночним комп'ютером, розбиваючи завдання на паралельно виконуються гілки, які обмінюються даними по зв'язує мережі. Одна з типових конфігурацій - набір комп'ютерів, зібраних із загальнодоступних компонентів, з встановленої на

них операційною системою Linux, і пов'язаних мережею Ethernet, Myrinet, InfiniBand або іншими відносно недорогими мережами [12].

ВИСНОВКИ

Кластерні обчислення надають ряд переваг щодо звичайних паралельних комп'ютерів, виготовлених на замовлення, для досягнення продуктивності, більшої, ніж типова для уніпроцесорів. Як наслідок, поява кластерів значно розширила доступність високопродуктивної обробки для набагато ширшого співтовариства та посилила її вплив завдяки новим можливостям в галузі науки, техніки, промисловості, медицини, комерції, фінансів, оборони та освіти серед інших секторів обчислювального застосування. Серед найважливіших переваг кластерних обчислень є:

– масштабованість продуктивності. Кластеризація комп'ютерних вузлів забезпечує засоби збирання більших систем, ніж це практично для нестандартних паралельних систем, оскільки вони самі можуть стати вузлами кластерів;

– продуктивність за вартістю. Кластеризація масових комп’ютерних систем дає перевагу ринку набагато ширшу, ніж обмежена для високопродуктивного обчислювального співтовариства. Для багатьох застосувань досягається перевага в ціновому відношенні в порівнянні із спеціально розробленими паралельними комп’ютерами;

– гнучкість конфігурації. Організація кластерних систем визначається топологією їх мереж взаємозв'язку, яка може бути визначена під час встановлення та легко модифікована. Залежно від вимог користувацьких програм, можуть бути реалізовані різні конфігурації системи для оптимізації пропускної здатності та затримки потоку даних;

– простота оновлення. Старі компоненти можуть бути замінені або нові елементи додані до оригінального кластера, щоб поступово покращити роботу системи, зберігаючи більшу частину початкових вкладень в апаратне та програмне забезпечення.

– конвергенція архітектури. Кластерні обчислення пропонують єдину загальну стратегію впровадження та застосування паралельних високопродуктивних систем, незалежних від конкретних постачальників обладнання та їх рішень щодо продукту. Користувачі кластерів можуть будувати системи програмних додатків із впевненістю, що такі системи будуть доступні для їх підтримки в довгостроковій перспективі;

– відстеження технологій. Кластери забезпечують найшвидший шлях до інтеграції найновіших технологій високопродуктивних обчислень, оскільки досягнення технологій пристроїв зазвичай спочатку включаються в комп'ютери масового ринку, придатні для кластеризації;

– висока доступність. Кластери надають кілька зайвих однакових ресурсів, які при правильному управлінні можуть забезпечити безперервну роботу системи завдяки витонченій деградації, навіть якщо окремі компоненти виходять з ладу.

Переваги операційної системи Linux:

– безкоштовність. Ядро Linux і основні компоненти, з яких складається система, і безліч програм поширюються з відкритим вихідним кодом абсолютно безкоштовно;

– настроюваність. З огляду на відкритий вихідний код, при наявності певних знань ви можете змінити в системі все, що завгодно, і так, як вам захочеться;

– простота встановлювання. Популярні дистрибутиви Linux дуже прості в установці. Ту ж Ubuntu можна запустити з флешки без установки і протестувати практично всі можливості операційної системи;

– безпека. Через низьку популярність Linux для робочих столів і архітектури системи, зловити вірус в Linux досить складно;

– драйвери обладнання. Ядро Linux містить всі вільні драйвери обладнання, з яким може працювати Linux;

– зручна система зберігання налаштувань. У Windows всі налаштування зберігаються в реєстрі.

Недоліки операційної системи Linux:

– складність засвоєння. Як би там не було, Linux дуже сильно відрізняється від Windows, тому спочатку буде складно його освоювати і звикати до нової концепції операційної системи;

– відсутність версій популярних програм. Це основний недолік, через який багато користувачів все ще не можуть повністю перейти на Linux.

На підставі вище викладеного вважаю що мета дипломної роботи досягнута в повній мірі.

ПЕРЕЛІК ПОСИЛАНЬ

1. http://icybcluster.org.ua/index.php?lang_id=2&menu_id=6
2. http://rsusu1.rnd.runnet.ru/tutor/method/m1/page06.html
3. https://uk.wikipedia.org/wiki/Комп%27ютерний_кластер
4. https://cluster.linux-ekb.info/mpi.php
5. https://www.sciencedirect.com/topics/computer-science/cluster-computing
6. https://uk.wikipedia.org/wiki/Цифрове_моделювання
7. https://ravesli.com/chto-takoe-linux-ego-struktura-i-preimushhestva/
8. https://www.linuxadictos.com/uk/найкращі-дистрибутиви-gnu-linux-2020.html
9. https://losst.ru/plyusy-i-minusy-linux
10. http://www.vsemcomp.ru/articles/61/
11. https://ko.com.ua/primer_postroeniya_otkazoustojchivogo_klastera_126576
12. https://newtravelers.ru/uk/iphoneipad/klasternaya-operacionnaya-sistema-referat-klasternye-sistemy-setevoe.html