МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

РОЗПОДІЛЕНІ СХОВИЩА ДАНИХ

Методичні рекомендації
до виконання лабораторних робіт
для студентів спеціальності
122 "Комп'ютерні науки та інформаційні технології"
другого (магістерського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2017

УДК 004.65(07)
ББК 32.973–018.2р
Р 65

Затверджено на засіданні кафедри інформаційних систем.
Протокол № 2 від 22.09.2016 р.

Самостійне електронне текстове мережеве видання

Укладач В. О. Алексієв

Р 65 **Розподілені** сховища даних : методичні рекомендації до виконання лабораторних робіт для студентів спеціальності 122 "Комп'ютерні науки та інформаційні технології" другого (магістерського) рівня : [Електронне видання] / уклад. В. О. Алексієв. – Харків : ХНЕУ ім. С. Кузнеця, 2017. – 66 с.

Наведено приклади та практичні завдання щодо виконання лабораторних робіт. Розглянуто основи побудови розподілених файлових сховищ даних та технології розподілених баз даних. Досліджено принципи побудови WEB-сервісів, що застосовують технології розподілених сховищ даних.

Рекомендовано для студентів спеціальності 122 "Комп'ютерні науки та інформаційні технології" другого (магістерського) рівня.

УДК 004.65(07)
ББК 32.973–018.2р

Вступ

Ефективне функціонування та конкурентоспроможність для багатьох сучасних підприємств пов'язані із обробкою великих обсягів даних. Для збереження яких слід залучати чи організувати спеціалізовані сховища, що мають властивості масштабування, надійного збереження інформації та зручності програмного інтерфейсу (API – *Application programming interface*). Можна навести багато прикладів застосування розподілених сховищ даних, наприклад, сучасні технології Інтернету речей (IoT – *Internet of Things*), що впроваджуються все більше до повсякденної діяльності мешканців міст та регіонів, стають основою для проектів "розумного будинку" чи "розумного міста" і потребують надійних та зручних технологій збереження даних, як підґрунтя для впровадження складних інтелектуальних систем.

Слід відзначити, що для проведення наукових досліджень і розроблень майже у будь-якій галузі, актуальною є проблема оброблення значних структурованих і неструктурованих обсягів даних. Для збереження цих даних також є потреба в організації файлових сховищ даних або, навпаки, залучення масштабованих рішень реляційних чи нереляційних систем баз даних. При цьому слід відрізняти випадки, коли є відсутньою потреба у розподілених сховищах даних й слід застосовувати звичайні рішення рівня серверу чи робочої станції. Тому тематика лабораторних робіт з дисципліни, що базується на виконані типових завдань у середовищі сучасних віртуальних систем, дозволяє студентам оцінити рівень складності певних рішень та визначити їх місце у складі систем обробки даних та прийняття рішень.

Навчальна дисципліна "Розподілені сховища даних" (РСД) є базовою навчальною дисципліною та вивчається згідно з навчальним планом підготовки фахівців освітнього ступеня "магістр" спеціальності 122 "Комп'ютерні науки та інформаційні технології" другого (магістерського) рівня. Методичні рекомендації до виконання лабораторних робіт за цією дисципліною ґрунтуються на кращих практиках сучасного розвитку технологій розподіленого збереження даних та систем масштабування відповідних серверних додатків, що є, у свою чергу, базовим рівнем побудови систем хмарних обчислень (*Cloud Computing*) [1 – 13; 79–82].

Лабораторне заняття – це вид навчального заняття, на якому студенти під керівництвом викладача індивідуально проводять імітаційні дослідження щодо проектування інформаційних систем (ІС) на базі застосування

технологій розподілених сховищ даних. Розгортання та налагодження систем доцільно виконувати у середовищі віртуалізації *Oracle VM VirtualBox* (https://www.virtualbox.org/) або *VMware Workstation Player* (http://www.vmware.com/products/player/playerpro-evaluation.html) чи інші, включно сервіси хмарних обчислень, що надають можливості моделювання мережевої структури розподіленого обчислювального середовища. У ході лабораторних занять студент набуває професійних компетентностей та практичних навичок роботи з сучасними програмними продуктами щодо розгортання, адміністрування та застосування технологій розподілених сховищ даних. Також у студентів формуються навички самостійності під час прийняття рішень для проектування та налагодження складних комп'ютерних систем.

Метою проведення лабораторних занять з навчальної дисципліни є отримання студентами практичних навичок роботи з програмними пакетами щодо виконання завдань розгортання, налагодження та адміністрування складних ІС на базі розподілених сховищ даних.

Об'єктом лабораторних занять з навчальної дисципліни є технологічні процеси розгортання, налагодження та супроводження ІС, які базуються на застосуванні розподілених сховищ даних.

Предметом лабораторних занять з навчальної дисципліни є розподілені сховища даних, включно розподілені файлові системи й розподілені бази даних на основі реляційної та нереляційної моделей.

Лабораторні заняття з навчальної дисципліни проводяться у спеціально оснащених обчислювальних центрах ХНЕУ ім. С. Кузнеця. Заняття починається зі стислого вступу викладача, в якому оголошується тема та цілі лабораторної роботи, вказівки щодо роботи з лабораторним обладнанням та оформлення звіту з лабораторної роботи. На початку кожного лабораторного заняття проводиться первинний контроль знань щодо готовності студентів до виконання лабораторної роботи.

За результатами виконання завдання на лабораторному занятті студенти формують звіт у електронному вигляді, у форматі сумісному з *Microsoft Word* "**.docx*" й надсилають на зазначену електронну адресу та захищають їх перед викладачем. Електронний звіт із лабораторної роботи повинен містити хід виконання роботи та стислий зміст теоретичних знань, які отримані студентом.

У звіті обов'язково наводяться основні скріншоти екранів, які свідчать про успішне виконання завдання або відбивають складності, що виникли під час виконання певних завдань у поточному вигляді із актуальними на

час виконання лабораторної роботи версіями програмних продуктів. Звіт із лабораторної роботи повинен бути оформлений відповідно до ДСТУ 3008-95 "Державний стандарт України. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення" та мати висновки щодо результатів виконання певної лабораторної роботи.

Також до звіту із виконання лабораторної роботи слід додавати посилання на літературні джерела та WEB-ресурси, що були застосовані у ході виконання лабораторної роботи.

Лабораторні заняття з навчальної дисципліни "Розподілені сховища даних" базуються на застосуванні відкритих операційних систем на основі ядра Linux, наприклад, застосовуються дистрибутиви *Ubuntu* або *CentOS*. Передбачається, що студенти в змозі розгорнути відповідні віртуальні машини [14; 18; 20; 27; 29; 31 – 33; 40; 42 – 47; 54; 55; 63; 73; 74]. Процес налагодження віртуальної машини на базі обраних дистрибутивів Linux стисло розглянуто у додатках.

Для формування віртуального середовища рекомендується застосовувати віртуальні машини, що мають ресурси: однин процесор, 1 Гб – оперативної пам'яті (мінімум 512 Мб), 8 Гб – накопичувач (віртуальний диск може мати меншу ємність до 4 Гб, однак, можна рекомендувати застосовувати режим динамічного розширення диску та обирати максимальний розмір порядку 20 Гб). Мережевий інтерфейс зручніше конфігурувати як *Bridged Networking* (для виконання лабораторних робіт можна застосувати DHCP-сервіс навчальної мережі, однак, у промислових умовах бажано, щоб кожен сервер мав відокремлену фіксовану IP-адресу). Для другого мережевого інтерфейсу можна обрати режим внутрішньої віртуальної мережі. Таким чином – один мережевий інтерфейс забезпечить можливість встановлення програмного забезпечення з Інтернет, а другий – комунікацію, наприклад, у межах віртуального кластеру.

Лабораторна робота 1
Організація сховищ даних типу SAN та NAS на основі дистрибутиву FreeNAS

Мета лабораторної роботи

Ознайомитися з технологіями на базі операційних систем *Linux* та *FreeBSD* щодо роботи із локальними дисковими системами, масштабуванням цих рішень і базовими мережевими протоколами, що забезпечують віддалений доступ до сховищ даних.

Рекомендації щодо підготовки до виконання лабораторної роботи

1. Ознайомитися із загальними відомостями про системи збереження даних, звернути особливу увагу на технології NAS та SAN [22; 49].

Слід зазначити, що мережева система зберігання даних або мережеве сховище даних (NAS – *Network Attached Storage*) є комп'ютером, сервером або кластером, що має дискові накопичувачі, які підключені до мережі (переважно локальної) частіше за протоколами NFS, SMB/CIFS та ін. Звичайно диски у такій NAS-системі поєднують до RAID-масивів. Така архітектура надає можливості створити надійні, масштабовані мережеві сховища даних, що мають зручні засоби для їх адміністрування (за матеріалами з Вікіпедії).

У свою чергу, мережа зберігання даних (SAN – *Storage Area Net-work*) є архітектурним рішенням для підключення зовнішніх пристроїв зберігання даних, таких як дискові масиви, стрічкові бібліотеки, оптичні накопичувачі, до серверів у такий спосіб, щоб операційна система розпізнала підключені ресурси як локальні. У визначених системах переважно використовується протокол iSCSI, який надає можливості створення бездискових клієнтських систем. Сервер чи вузол, який не має дискових накопичувачів може виконати завантаження у мережі TCP/IP із накопичувача, який є фізичною частиною дискового масиву SAN-системи (LUN – *Logical Unit Number*).

2. Для подальшої роботи у середовищі операційної системи Linux доцільним є ознайомлення із особливостями технології LVM [28].

3. Виконати оцінку перспектив застосування інтерфейсу iSCSI для побудови систем типу SAN [15; 50; 51; 56].

4. Визначитися із реалізаціями RAID-масивів. Приділити увагу сховищам даних на основі технології файлової системи ZFS та масивів RAID-Z [19; 21; 48].

5. Ознайомитися із основними вимогами дистрибутиву FreeNAS до апаратного забезпечення вузла [67 – 70].

Слід відзначити, що дистрибутив *FreeNAS* (http://www.freenas.org/) оснований на операційній системі *FreeBSD* та спрямований на рішення завдань організації сховищ даних типу SAN та NAS. *FreeNAS* має підтримку поширених протоколів доступу до мережевих сховищ даних, наприклад, SMB/CIFS (доступ до мережевих ресурсів віддалених папок у операційній системі *Windows*), NFS (*Unix*-подібні мережеві файлові ресурси) та AFP (доступ до файлових сховищ даних у мережі пристроїв *Apple*), а також розповсюджених протоколів універсального призначення FTP, iSCSI, *WebDAV* та ін.

Спрощена структура системи FreeNAS фактично реалізує виконання завдань щодо адміністрування FreeBSD завдяки застосуванню зручного WEB-інтерфейсу системи (рис. 1).

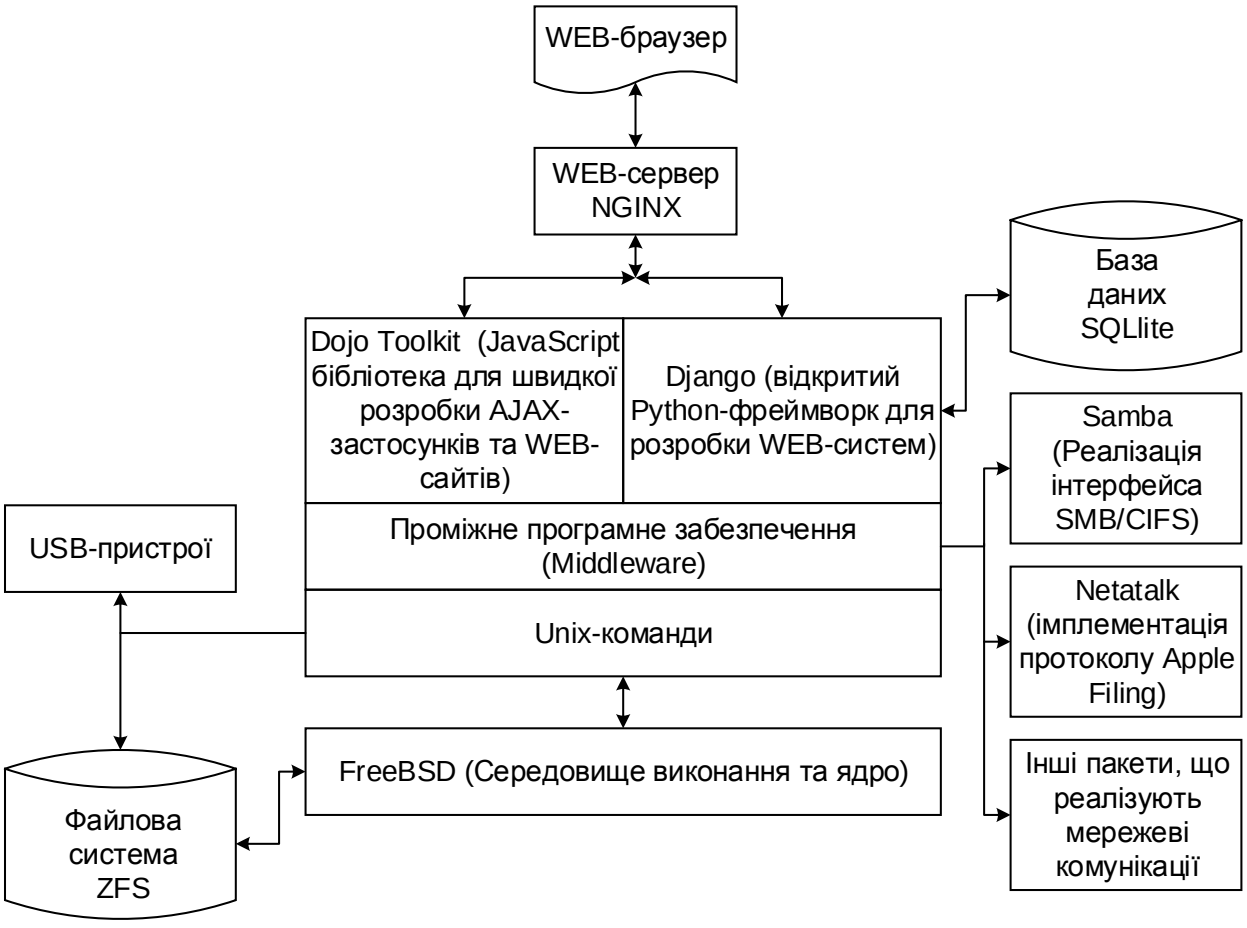

Рис. 1. **Спрощена архітектура системи FreeNAS 9**

Система *FreeNAS* ґрунтується на файловій системі ZFS та відповідному рішенні сховища типу RAID-Z [75]. Однак, користувач системи завдяки

зручному WEB-інтерфейсу завжди може обрати іншу реалізацію файлової системи та організацію дискових масивів. Основним у такому виборі є фактична підтримка операційною системою FreeBSD відповідних технологій збереження даних та мережевих технологій організації віддаленого доступу. Завдяки технології розширення системи *FreeNAS* можна значно вдосконалити рішення SAN чи NAS, що розгортається, наприклад, за допомогою плагінів мережевої системи резервування даних *bacula* або розгортання приватного хмарного середовища доступу до файлових ресурсів за допомогою *ownCloud* та ін.

Порядок виконання лабораторної роботи:

1. У середовищі віртуалізації, наприклад, *VirtualBox* або *VMware Player* виконати установку *FreeNAS* (рис. 2). Параметри віртуальної машини: оперативна пам'ять не менше 1 Гб (рекомендовано розробниками 8 Гб); HDD – бажано застосувати три накопичувача, кожен не менш ніж 4 Гб; мережевий інтерфейс: *Bridget*.

Для доступу до WEB-інтерфейсу системи *FreeNAS*, яку було розгорнуто як віртуальну машину, доцільно застосовувати WEB-браузер основної операційної системи, у середовищі якої виконується відповідна система віртуалізації. Для тестування віддаленого доступу до ресурсів файлового сховища доцільно додатково розгорнути віртуальні машини на базі *Linux*-дистрибутиву *Ubuntu* та *Microsoft Windows Server*.

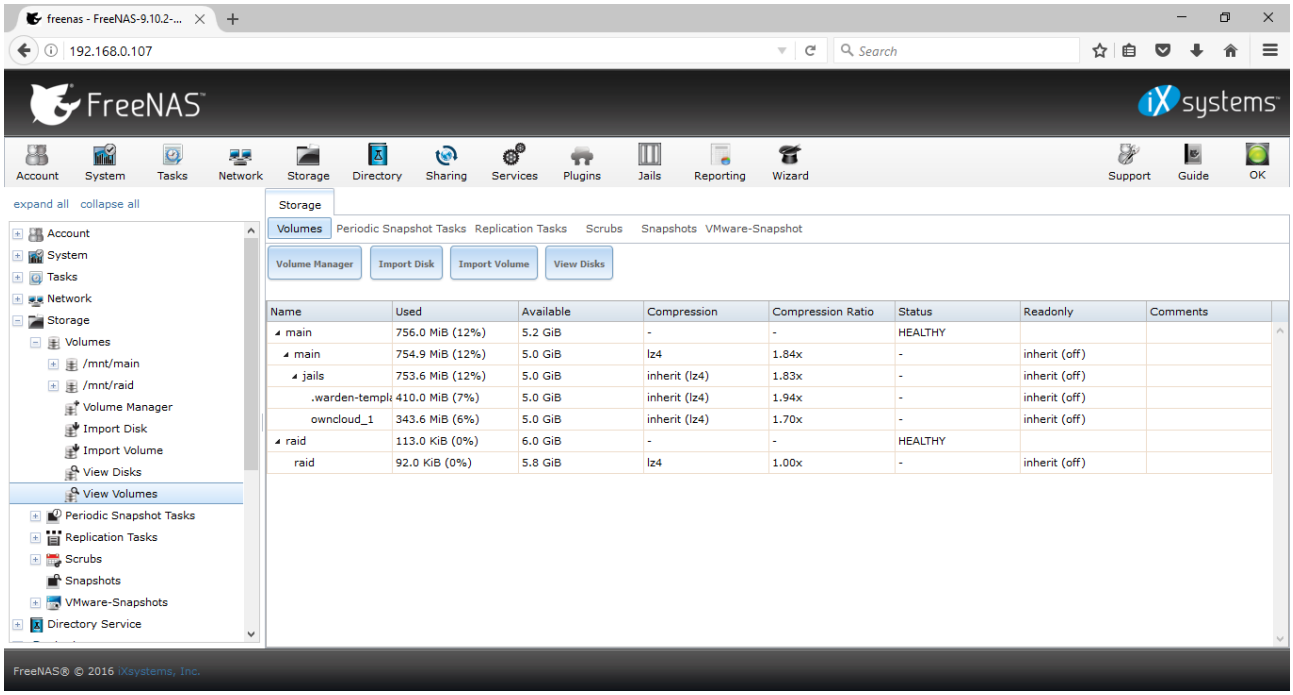

Рис. 2. **WEB-інтерфейс системи *FreeNAS***

2. Виконати настройку FreeNAS: додати дисковий масив у форматі RAID із повним віддзеркалюванням. Надати віддалений доступ до ресурсів файлового сховища за протоколами: CIFS, NFS, FTP та WebDAV. У середовищі віртуальної машини на базі дистрибутиву Linux виконати настройку доступу до файлового сховища за технологіями NFS та FTP. У середовищі операційної системи Microsoft Windows Server організувати доступ до віддалених ресурсів файлового сховища за технологіями: CIFS, FTP та WebDAV.

Для роботи із протоколом FTP у середовищі операційної системи *Microsoft Windows* доцільно застосовувати програмне забезпечення *FileZilla Client*. Настройку доступу за протоколом NFS у середовищі *Linux* можна виконати за допомогою команди монтування розділу файлової системи *mount*. Для доступу за технологією *WebDAV* у операційних системах типу *Windows* слід застосовувати вмонтовані засоби.

Протягом роботи із мережевими системами доступу до серверу NAS слід виконати моделювання стану поломки одного із дисків RAID-масиву. Для цього у віртуальній машині, із встановленим дистрибутивом *FreeNAS*, слід відключити один із дискових накопичувачів, на базі яких розгорнуто програмний RAID-масив. Потім слід повернути попередню конфігурацію із новим віртуальним диском схожим за розміром із видаленим. Після додавання диску слід переконатися у тому, що файлове сховище знову працює у штатному режимі.

3. Додатково можна виконати розгортання сховища даних типу SAN на базі технології iSCSI. Для перевірки доступу до сховища даних слід застосувати віртуальну машину із встановленою операційною системою типу *Microsoft Windows Server* та засоби ініціатору iSCSI. Також для роботи із технологією iSCSI можна застосовувати інші клієнтські машини на базі операційної системи Windows [58; 71].

Після закінчення виконання лабораторної роботи слід зупинити всі залучені віртуальні машини та видалили їх файли з жорсткого диску комп'ютера на навчальному робочому місці.

Завдання до лабораторної роботи:

1. Ознайомитися з подібними до *FreeNAS* системами, наприклад, *NAS4Free* (http://www.nas4free.org/) на базі операційної системи *Linux*. Визначити характерні ознаки, особливості розгортання та застосування їх у промислових умовах. Визначити відповідно до рекомендацій щодо

порядку проведення лабораторної роботи та наявного комп'ютерного обладнання навчальної лабораторії обсяги проведення та виконання завдань відповідної лабораторної роботи. Наприклад, у разі неможливості розгортання у середовищі віртуалізації системи на базі *FreeBSD* застосувати обране за схожими властивостями рішення на базі операційної системи *Linux*.

2. Виконати розгортання та налагодження *FreeNAS* чи аналогічного рішення на ресурсах окремої віртуальної машини.

3. Розгорнути віртуальну машину із дистрибутивом *Ubuntu* та налаштувати її мережеві інтерфейси відповідно до "Інструкції користувача", розділу (http://help.ubuntu.ru/wiki/настройка_сети_вручную).

4. За рекомендаціями [41; 72] та *How To Configure RAID Arrays on Ubuntu* 16.04 користувачів сервісу *DigitalOcean* (https:// www.digitalocean.com/ community/tutorial_series/how-to-configure-raid-arrays-on-ubuntu-16-04) виконати дії щодо створення програмного RAID-масиву типу дзеркало RAID 1.

Визначити наявні дискові накопичувачі:

lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT

Поєднати до масива два накопичувача, наприклад, *sda* та *sdb*:

sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sda /dev/sdb

Створити файлову систему на новому масиві:

sudo mkfs.ext4 -F /dev/md0

Створити точку монтування файлової системи та додати її до наявних ресурсів:

sudo mkdir -p /mnt/md0

sudo mount /dev/md0 /mnt/md0

Потім слід додати до конфігураційного файла */etc/mdadm/mdadm.*conf утіліти *mdadm* дані про створену конфігурацію RAID та зберегти у файлі */etc/fstab* конфігурацію файлової системи, що було додано до загальних ресурсів системи.

Слід зазначити, що у разі розгортання операційної системи, наприклад, *Ubuntu*, засоби установки цієї системи дозволяють у автоматичному режимі залучити програмний RAID-масив для розгортання відповідного програмного продукту але це не є предметом лабораторної роботи, що розглядається.

5. На базі дистрибутиву *Ubuntu* за рекомендаціями *How To Install and Configure ownCloud on Ubuntu* 16.04 користувачів сервісу *DigitalOcean* (https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-

owncloud-on-ubuntu-16-04) або у якості розширення до *FreeNAS* ознайомитися із можливостями системи *ownCloud* (https://owncloud.org/).

6. Виконати оформлення звіту щодо результатів виконання завдань лабораторної роботи.

Контрольні питання:

1. Назвіть переваги застосування RAID-масивів. Чим відрізняється RAID 0, RAID 1 та RAID 5? Наведіть особливості застосування програмних та апаратних RAID-систем.

2. Що таке файлова система у сенсі визначення технологій рівня операційної системи?

3. Чим відрізняються технології SAN та NAS? Наведіть приклади.

4. Назвіть переваги та недоліки застосування спеціалізованих дистрибутивів для організації сховищ даних на прикладі дистрибутиву FreeNAS.

5. Наведіть визначення термінів CIFS, NFS, FTP та *WebDAV*. Поясніть як застосовуються відповідні протоколи доступу до сховищ даних у системах на базі *Linux* та *Windows*.

Лабораторна робота 2
Налаштування розподіленої файлової системи *Ceph* в операційній системі *Linux* на базі віртуальних машин

Мета лабораторної роботи

Визначити особливості застосування промислових розподілених сховищ даних. Ознайомитися із принципами та засобами розгортання й супроводження розподілених файлових систем.

Рекомендації щодо підготовки до виконання лабораторної роботи

Ceph – розподілена файлова система з відкритим вихідним кодом, яка масштабуються до петабайтних розмірів, забезпечуючи зберігання і реплікацію даних, а також розподіл навантаження, що гарантує високу доступність і надійність збереження даних [25; 46; 60; 61]. Під час виходу

будь-якого диска, вузла або групи вузлів з ладу, технології *Ceph*, що базуються на дублюванні даних, забезпечать прозоре для користувачів відновлення втрачених копій на інших вузлах. На рис. 3 зображено спрощену архітектуру складових кластера *Ceph* (http://docs.ceph.com/ docs/ master/ architecture/).

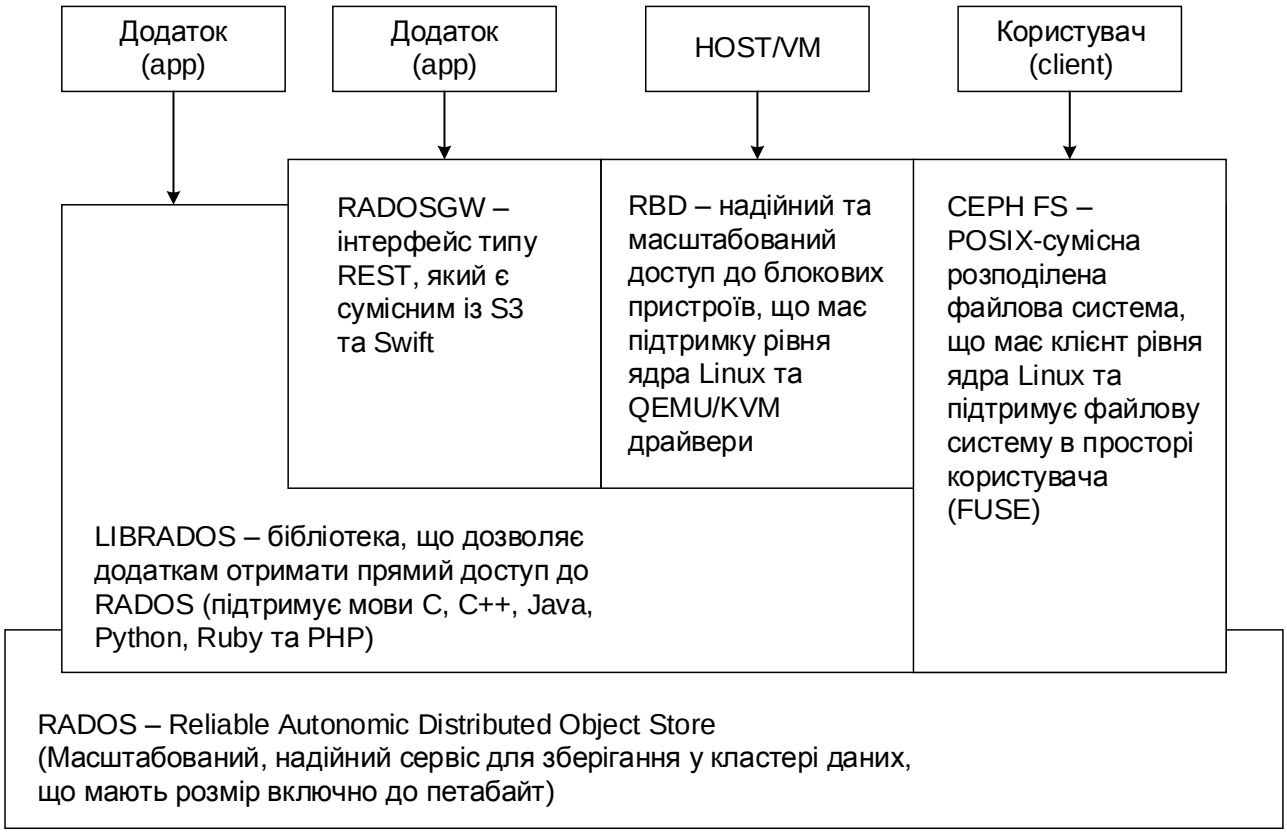

Рис. 3. **Архітектура системи *Ceph***

Ceph надає на вибір три різні абстракції для роботи зі сховищем: абстракцію об'єктного сховища (RADOS *Gateway*), блокового пристрою (RADOS *Block Device*) або POSIX-сумісної файлової системи (*CephFS*). Абстракція об'єктного сховища поруч із *FastCGI*-сервером дозволяє використовувати *Ceph* для зберігання призначених для користувача об'єктів та надає API, сумісний з *S3* / *Swift*. Абстракція блочного пристрою надає користувачам можливість створювати і використовувати віртуальні блокові пристрої довільного розміру. Програмний інтерфейс RBD дозволяє працювати з цими пристроями в режимі читання/запису і виконувати службові операції – зміна розміру, клонування, створення і повернення до знімка стану та ін. Гіпервізор *QEMU* містить драйвер для роботи з *Ceph* і забезпечує віртуальним машинам доступ до блокових пристроїв RBD. Тому *Ceph* зараз

підтримується у всіх популярних хмарних рішеннях – *OpenStack*, *CloudStack*, *Proxmox*. Файлова система *CephFS* використовується у ролі сховища даних, однак, на теперішній час не є готовою до виробничого застосування завдяки частково нереалізованим можливостям оновлення після збоїв.

Завдяки бібліотекам *librbd*, *librgw* та *libcephfs* клієнтські системи можуть взаємодіяти із відповідними складовими кластера *Ceph* (рис. 4). Крім того, завдяки інтерфейсу *librados* API користувачі можуть розробляти програмні рішення щодо взаємодії з *Ceph Storage Cluster.*

<table>
<tr><td>Блокові пристрої (Block Device)</td><td>Об'єктне сховище (Object Storage)</td><td>Кластерна файлова система (CephFS)</td></tr>
<tr><td>Бібліотека librbd</td><td>Бібліотека librgw</td><td>Бібліотека libceph</td></tr>
<tr><td colspan="3">Протокол кластера Ceph Storage Cluster Protocol (librados)</td></tr>
<tr><td>Сервіси об'єктів зберігання (OSDs)</td><td>Сервіси метаданих (MDSs)</td><td>Монітори (Monitors)</td></tr>
</table>

Рис. 4. **Структура складових системи *Ceph***

Оскільки система *Ceph* є програмним рішенням, що працює над рівнем стандартних файлових систем та мережевих протоколів, для побудови відповідного гетерогенного кластера можна задіяти різні сервери, які оснащені дисковими накопичувачами, що також можуть мати різний розмір. За управлінням операційної системи працюють сервіси *Ceph*, які виконують різні ролі кластера.

Сервіс (демон) об'єкта зберігання (*Ceph Object Storage Device Daemon*, OSD) – сутність, яка відповідає за зберігання даних, та є основною складовою кластера *Ceph*. На одному фізичному сервері може розміщуватися кілька OSD, кожен з яких має керувати окремим фізичним сховищем даних (накопичувачем). Сервер метаданих (*Ceph Metadata Server,* MDS) є допоміжним сервісом, що забезпечує синхронізацію стану файлів у точках монтування *CephFS*. Монітор (*Ceph Monitor,* MON) – елемент інфраструктури *Ceph*, який забезпечує адресацію даних усередині кластера і зберігає інформацію про топологію, стан та розподілі даних усередині сховища. Монітор виконує роль координатора, з якого починається кластер. Як тільки

відбувається розгортання хоча б одного монітору, можна говорити про запуск *Ceph*-кластера. Клієнти звертаються до моніторів, щоб дізнатися, на які OSD писати/читати дані. Бажано в кластері розгорнути декілька моніторів, для того щоб виключити єдину точку відмови системи.

Слід відзначити, що загалом один сервер може виступати, наприклад, як в ролі монітора (MON), так й в ролі сховища даних (OSD). Інший сервер може поєднувати ролі сховища даних та сервера метаданих (MDS). У великих кластерах сервіси запускаються на окремих машинах, але в невеликих рішеннях, де кількість серверів значно обмежена, деякі сервери можуть виконувати відразу декілька завдань кластера.

Порядок виконання лабораторної роботи

Для дослідження роботи системи *Ceph* доцільно виконати розгортання системи в умовах застосування системи віртуалізації. Найпростішим рішенням є застосування трьох вузлів, на яких слід встановити відповідні компоненти OSD та MON кластера *Ceph* [76; 77].

Слід відзначити, що оптимальною є конфігурація, яка передбачає забезпечення роботи вузла кластера двома мережевими інтерфейсами. Один з яких призначено для роботи у мережі Інтернет, наприклад, для оновлення операційної системи вузла, а інший інтерфейс слід налаштувати для взаємодії вузлів кластера (рис. 5).

Для розгортання кластера *Ceph* доцільно скористатися інструментом *ceph-deploy* для установки та налаштування *ceph* на всіх трьох віртуальних машинах. Також слід виконати налаштування вузлів для забезпечення взаємодії за протоколом ssh без пароля. Для цього слід застосувати команду: *ssh-keygen.* Також у середовищі операційної системи *Ubuntu* зазвичай не активовано користувача *root*, тому слід активувати відповідний обліковий запис, для чого вказати новий пароль супер-користувача: *passwd root*. Для генерації ключів шифрування не слід вказувати фразу-пароль, для того щоб не вводити відповідний пароль для виконання дій із ключем шифрування на вузлах. Наступним кроком слід застосувати команду, наприклад: *ssh-copy-id ceph-node2* для копіювання ключів шифрування до відповідних вузлів.

Наступним кроком стає встановлення утиліти *ceph-deploy* на всі вузли кластера та за її допомогою розгортання відповідних програмних складових системи *Ceph*.

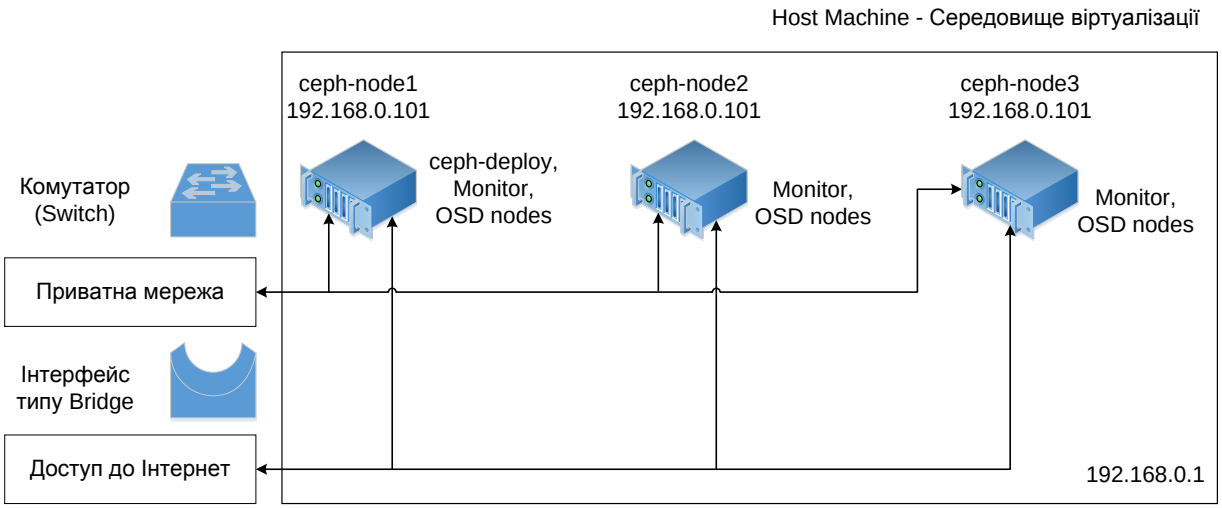

Рис. 5. **Топологія мережі *Ceph*-кластера в середовищі віртуалізації**

Після закінчення виконання лабораторної роботи слід зупинити всі залучені віртуальні машини та видалили їх файли з жорсткого диску комп'ютера на навчальному робочому місці.

Завдання до лабораторної роботи:

1. Розгорнути три вузла на базі операційної системи *Ubuntu* у середовищі віртуалізації.

2. Встановити на кожному вузлі утиліту *ceph-deploy*. Для цього слід спочатку додати ключ версії:

wget -q -O- 'https://ceph.com/git/?p=ceph.git;a=blob_plain; f=keys/release.asc' | sudo apt-key add -

Додати відповідні репозиторії *Ceph*:

echo deb http://ceph.com/debian-{ceph-stable-release}/ $(lsb_release -sc) main | sudo tee /etc/apt/sources.list.d/ceph.list

Потім слід оновити поточні репозиторії та встановити *ceph-deploy*:

sudo apt-get update && sudo apt-get install ceph-deploy

3. Створити MON на потрібних вузлах за допомогою команди:

ceph-deploy mon create node01 node02 node03

Розгорнути OSD на вузлах, наприклад:

ceph-deploy osd create node01:sdb node01:sdc

ceph-deploy osd prepare node02:sdb node02:sdc

4. Виконати перевірку працездатності кластера за допомогою команди: *ceph -s* та переконатися, що кластер функціонує.

5. Оформити звіт з лабораторної роботи.

Контрольні питання:

1. Що таке розподілена файлова система? Наведіть приклади.

2. Що таке *Ceph*? Для чого застосовується ця система? Наведіть аналогічні рішення, в чому їх переваги чи недоліки?

3. Наведіть типові приклади застосування технологій розподілених сховищ даних.

4. Наведіть приклади реалізації технології прозорого доступу серверу кластера чи вузла до всіх інших вузлів кластера розподіленого сховища даних.

5. Наведіть типову структуру кластера файлового сховища даних. Які типи вузлів використовуються у наведеній архітектурі?

Лабораторна робота 3
Розгортання СКБД *MySQL* та налаштування механізму реплікації даних із застосуванням технологій віртуалізації

Мета лабораторної роботи

Оволодіти навичками адміністрування системою керування базою даних (СКБД) *MySQL*. Визначити засоби масштабування, забезпечення безвідмовної роботи та доступності реляційної СКБД *MySQL*.

Рекомендації щодо підготовки до виконання лабораторної роботи

Ознайомитися із загальними відомостями про реплікацію реляційних баз даних [2; 3; 5; 7; 8; 18; 27].

Порядок виконання лабораторної роботи:

1. Виконати у середовищі віртуалізації на двох вузлах (віртуальних машинах) базову установку операційної системи *Ubuntu Server 14.04.3 LTS*. Розгортання серверної операційної системи слід виконувати відповідно до рекомендацій розділу *Installation* офіційного керівництва *Ubuntu Server 14.04 Guide* (https://help.ubuntu.com/lts/serverguide/installing-from-cd.html).

Параметри віртуальної машини мають бути такими або кращими за: однин процесор, 1 Гб – оперативної пам'яті, 8 Гб – накопичувач (віртуальний диск). Мережевий інтерфейс зручніше конфігурувати як *Bridged*

Networking (для проведення лабораторної роботи можна застосувати DHCP-сервіс навчальної мережі, однак слід зазначити, що у промислових умовах кожен сервер повинен мати відокремлену фіксовану IP-адресу).

Кожна віртуальна машина повинна містити базову установку серверної операційної системи, що не містить додаткових сервісів, крім *OpenSSH Server* для віддаленого доступу до віртуальної машини.

2. Виконати розгортання СКБД *MySQL* у середовищі віртуальних машин відповідно до *How to Install MySQL on Ubuntu 14.04* керівництва сервісу *Linode* (https://www.linode.com/docs/databases/mysql/how-to-install-mysql-on-ubuntu-14-04). Для цього слід оновити систему:

sudo apt-get update

sudo apt-get upgrade

Виконати установку СКБД: *sudo apt-get install mysql-server*

Слід відзначити, що у ході установки обов'язково слід вказати пароль суперкористувача баз даних *root* (рис. 6).

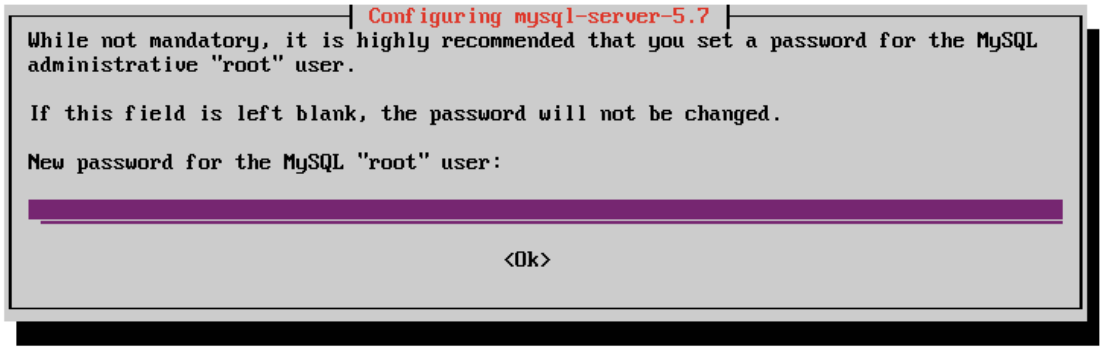

Рис. 6. **Установка СКБД *MySQL***

Для підвищення безпеки слід виконати скрипт, що забезпечує встановлення паролю суперкористувача (адміністратора) бази даних, якщо раніше не було встановлено пароля, також скрипт видаляє тестову базу даних: *mysql_secure_installation*

3. Налаштувати *Master*-сервер реплікації бази даних *MySQL* за рекомендаціями *How To Set Up Master Slave Replication in MySQL* користувачів сервісу *DigitalOcean* (https://www.digitalocean.com/community/tutorials/how-to-set-up-master-slave-replication-in-mysql). Для цього у файлі конфігурації СКБД *sudo nano /etc/mysql/my.cnf* вказати IP-адресу сервера у параметрі *bind-address* та встановити ідентифікатор сервера: *server-id = 1*. Також слід указати ім'я лог-файла реплікації: *log_bin = /var /log /mysql /mysql-bin.log*

та визначити ім'я бази даних, що буде брати участь у процесі реплікації: *binlog_do_db = newdatabase*. Для того щоб зміни почали діяти слід перезавантажити СКБД:

sudo service mysql restart

Потім слід скористатися клієнтським програмним забезпеченням СКБД *MySQL* та налаштувати права доступу для користувача, від імені якого будуть здійснюватися операції реплікації:

mysql -u root –p

*> GRANT REPLICATION SLAVE ON *.* TO 'slave_user'@'%' IDENTIFIED BY 'password';*

> FLUSH PRIVILEGES;

Для запуску репліки слід зробити дамп бази даних:

> USE newdatabase;

> FLUSH TABLES WITH READ LOCK;

> SHOW MASTER STATUS;

У таблиці, що відображає статус *Master*-сервера слід запам'ятати поточне значення стовпців: *File*, *Position*, що знадобиться під час налаштування *Slave*-сервера.

Зробити дамп бази даних слід командою:

mysqldump -u root -p --opt newdatabase > newdatabase.sql

Виконати розблокування бази даних:

> UNLOCK TABLES;

> QUIT;

4. Налаштувати роботу репліки. Спершу імпортувати базу даних із *Master*-сервера:

> CREATE DATABASE newdatabase;

> EXIT;

Також у командному рядку операційної системи слід виконати команду додавання на репліку копії бази даних, що була створена раніше на *Master*-сервері.

mysql -u root -p newdatabase < /path/to/newdatabase.sql

Потім виконати редагування файла конфігурації СКБД *MySQL*: *sudo nano/etc/mysql/my.cnf*. У цьому випадку слід указати ідентифікатор сервера: *server-id = 2* та вказати лог-файли:

relay-log =/var/log/mysql/mysql-relay-bin.log

log_bin = /var/log/mysql/mysql-bin.log

binlog_do_db = newdatabase

Для того щоб зміни у конфігурації *Slave*-сервера почали діяти слід перезавантажити СКБД:

sudo service mysql restart

Заключним кроком слід налаштувати параметри підключення до *Master*-сервера у середовищі клієнта сервера *MySQL*:

> CHANGE MASTER TO MASTER_HOST='IP_MASTER',

MASTER_USER='slave_user', MASTER_PASSWORD='password',

MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=107;

де *IP_MASTER* – IP-адреса *Master*-сервера; *MASTER_LOG_FILE*, *MASTER_LOG_POS* – значення, що відображали статус *Master*-сервера.

Активувати реплікацію та визначити статус репліки:

> START SLAVE;

> SHOW SLAVE STATUS\G

Після цього, зміни, що будуть відбуватися у структурі записів у базу даних Master-сервера, будуть дублюватися на сервері репліки.

Завдання до лабораторної роботи:

1. Виконати налаштування реплікації бази даних *MySQL* типу *Master-Slave* на двох вузлах, що функціонують на базі операційної системи *Ubuntu Server* 14.04.3 LTS або старшої.

2. Оформити звіт із лабораторної роботи.

3. Ознайомитися із особливостями реплікації та основними діями щодо вирішення ситуацій відмови *Master*-сервера за публікацією "Основы репликации в *MySQL*" на ресурсі Хабрахабр (http://habrahabr.ru/post/56702/).

4. Додатково ознайомитися із особливостями налаштування *Master-Master* реплікації за рекомендаціями *Configuring MySQL Master-Master Replication* сервісу *Linode* (https://www.linode.com/ docs/ databases/ mysql/ mysql-master-master-replication).

5. Скласти звіт із лабораторної роботи.

Контрольні питання:

1. Наведіть визначення реляційної бази даних. Що таке реплікація та сегментування бази даних? Наведіть приклади застосування.

2. Яким чином вирішуються завдання балансування навантаженням у системах розподілених сховищ даних?

3. Назвіть особливості реплікації БД типу *Master-Slave*.

4. Назвіть особливості реплікації БД типу *Master-Master*.

5. Назвіть відмінні особливості стратегії застосування розподілених сховищ даних для рішення бізнес-завдань великого Інтернет-магазина.

Лабораторна робота 4
Розгортання та налаштування кластера *Percona XtraDB Cluster* на базі технологій віртуалізації

Мета лабораторної роботи

Ознайомлення з технологіями розгортання кластера бази даних. Оволодіти навичками адміністрування рішення розподілених сховищ даних на базі СКБД *MySQL*. Ознайомитися із особливостями розгортання кластерних рішень баз даних.

Загальні тези лабораторної роботи

У багатьох сценаріях застосування у архітектурі сучасних додатків та WEB-рішень є бази даних. Проблеми масштабування баз даних поділяють, як правило, на два класи: за рішенням завдання забезпечення належної продуктивності та вимогами до СКБД щодо зберігання великої кількості даних. Знизити навантаження на базу даних можна за умови поширення її на декілька вузлів. Поруч з цим гостро постає проблема виконання синхронізації між вузлами.

У масштабуванні бази даних є свої нюанси, які враховують складність підтримки актуальності даних на багатьох серверах. Відповідно до цього оптимальна стратегія масштабування залежить від структури даних, архітектури і від типу проблеми.

У WEB-додатках, зазвичай, домінують запити на отримання інформації – читання коментарів, постів, призначених для користувача даних та ін. Таким чином, слабким місцем є операція читання, тому саме її потрібно масштабувати.

Для вирішення цього завдання використовується механізм реплікації – один сервер призначається головним (майстер-сервером), який виконує всі запити модифікації даних, а інші сервери (підлеглі) обробляють тільки запити отримання даних. У разі зміни або додаванні даних, майстер сервер повідомляє всі підлеглі (*slave*) сервера, таким чином підтримуючи актуальність даних. Крім того, реплікація використовується для географічного розподілу серверів [52; 26]. Така структура розглянута у попередній лабораторній роботі.

Майстер сервер під час виконання модифікацій записує всі зроблені зміни до журналу, а "*Slave*" сервера з деякою періодичністю перевіряють *Log* (журнал) на предмет появи нових даних і якщо вони є – виконують аналогічні дії зі своїми даними (рис. 7).

Така схема зазвичай використовується в додатках з домінуючими запитами на читання – всі запити на зміну бази направляються майстер сервера, тоді як запити на читання розподіляються між підлеглими.

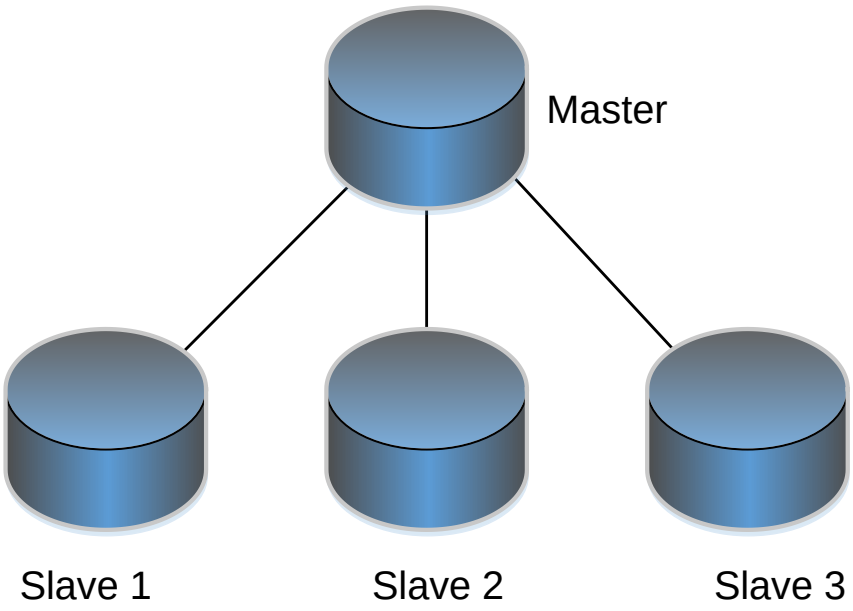

Рис. 7. **Масштабування бази даних "*Master-Slave*"**

Зазвичай у додатку вказується список серверів (пул) призначених для читання, з якого *MySQL*-клієнт певним чином (випадковим або за вказаною алгоритмом) вибирає сервер для виконання запиту, таким засобом балансуючи навантаження між усіма серверами. Недоліком схеми є ймовірність ситуації, за якою у разі виходу з ладу майстер сервера, всі запити на модифікацію і додавання даних не будуть виконуватися.

Дуже зручною для географічного розподілу даних є схема ланцюжка майстер серверів (рис. 8). У такій схемі забезпечується покращення доступності серверів СКБД за рахунок зменшення шляху подорожі запиту до сервера. Така схема є зручною у процесі масштабування бази даних, наприклад, складової інформаційного порталу тощо.

Недолік схеми аналогічний за попередній приклад – у разі виходу з ладу одного із майстер серверів, ланцюжок буде порушено, але регіональні сервера, що залишилися, будуть працювати незалежно.

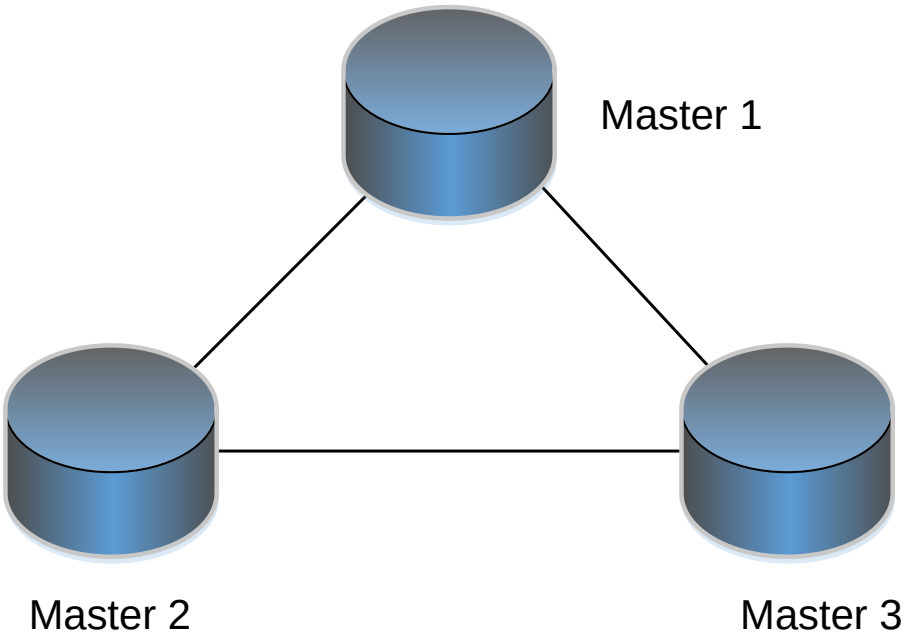

Рис. 8. **Масштабування бази даних "Master- Master"**

Схема ланцюжка з двох майстер серверів та багато підлеглих серверів дозволяє майстрам виконувати запити на модифікацію даних та мають рівну кількість підлеглих серверів. Таким чином, під час виходу з ладу одного майстра, додаток продовжить працювати з іншими підлеглими серверами.

Таким чином, рішення завдань масштабування та адаптації інформаційно-комунікаційної технології певного WEB-порталу до збільшення кількості користувачів на практиці зводиться до вертикального та, згодом, горизонтального нарощування потужності серверних ресурсів, поруч із визначенням оптимальної структури масштабування баз даних.

Відкритий продукт *Percona XtraDB Cluster* надає рішення для створення кластерів із синхронною реплікацією між вузлами, що працюють у режимі *multi-master*. Система базується на напрацюванні *Percona Server* і *Codership Galera Replicator*. *Percona XtraDB Cluster* забезпечує високу продуктивність, швидке відновлення вузла кластера після падіння і повний контроль стану кластера*. MariaDB Galera Cluster* – це *MariaDB* кластер із майстер-майстер реплікацією, який використовує для синхронізації *galera*-бібліотеку [53; 66].

Завдання до лабораторної роботи:

1. Виконати розгортання та налаштування кластера *Percona XtraDB Cluster* на базі технологій віртуалізації. За рекомендаціями офіціальної документації *Percona XtraDB Cluster* 5.7 (https://www.percona.com/doc/percona-

xtradb-cluster/5.7/index.html) на прикладі розгортання кластера із вузлами на базі операційної системи *Ubuntu Server.*

Виконуємо команду: *wget https://repo.percona.com/apt/percona-release_0.1-4.$(lsb_release -sc)_all.deb* за якою, із офіційного репозиторію *Percona*, буде завантажено пакет для розгортання кластера.

На цьому етапі є доцільним оновити залежності версій пакетів, які доступні у поточній установці операційної системи: *sudo apt-get update* та виконати оновлення програмного забезпечення: *sudo apt-get upgrade.*

Наступним кроком за допомогою пакетного менеджера з дистрибутиву, сумісного із *Debian*, виконаємо установку та налагодження пакетів, які знадобляться для розгортання кластера:

sudo dpkg -i percona-release_0.1-4.$(lsb_release -sc)_all.deb

Знову оновимо локальний кеш репозиторію: *sudo apt-get update*

Для розгортання вузлів кластеру СКБД *Percona XtraDB Cluster* слід застосувати пакетний менеджер операційної системи *Ubuntu*:

sudo apt-get install percona-xtradb-cluster-full-57

На рис. 9 наведено процес розгортання системи на окремому вузлі у терміналі *PuTTY* за технологією віддаленого доступу до вузла *ssh*.

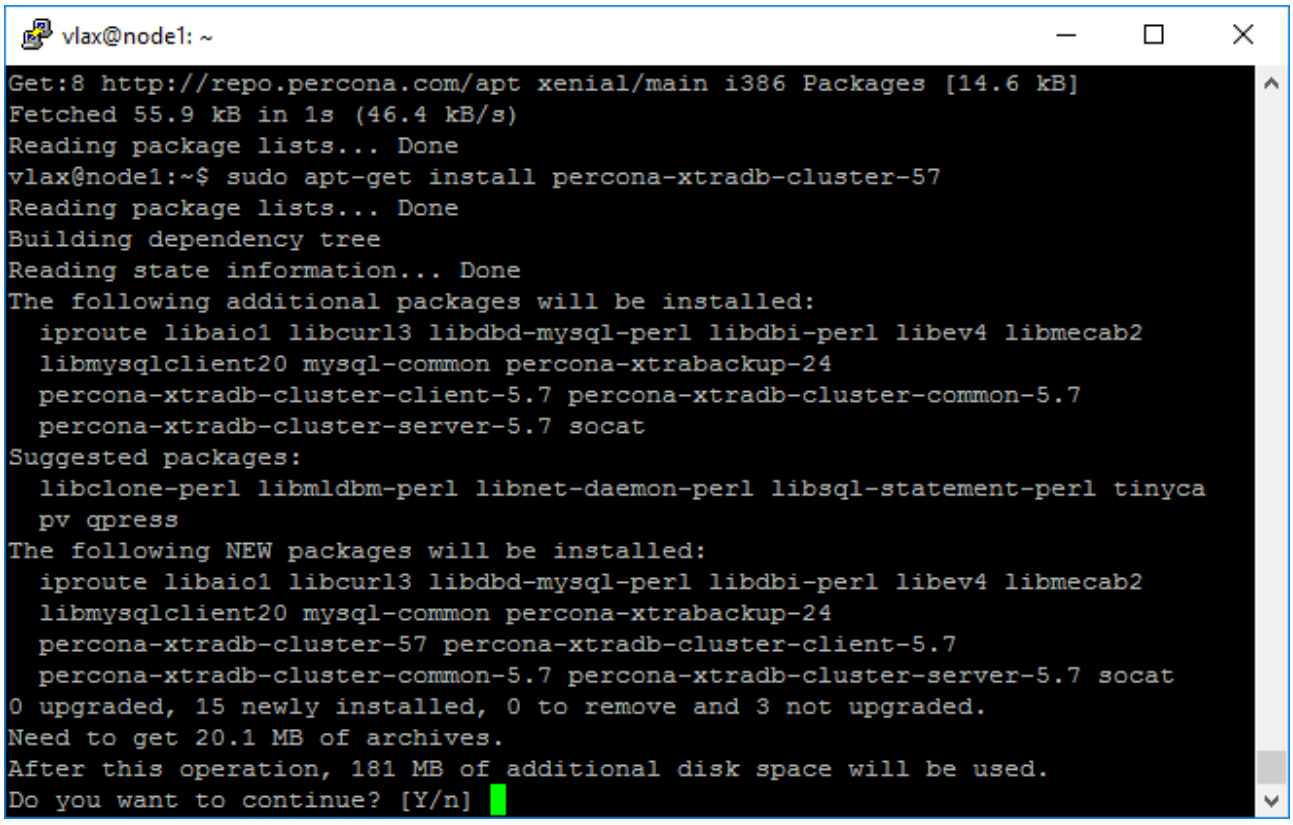

Рис. 9. **Розгортання вузла *Percona XtraDB Cluster***

Відповідно до звичайної установки СКБД *MySQL,* у ході розгортання вузла кластеру слід буде вказати пароль суперадміністратора (*root*) бази даних.

Слід зазначити, що для запобігання конфлікту версій СКБД, у разі розгортання *Percona XtraDB Cluster* на вузлі із встановленою раніше системою *MySQL* або *MariaDB*, доцільно спочатку виділити із системи попередню версію СКБД.

Наступним кроком розгортання кластеру на визначеному вузлі із встановленим програмним забезпеченням буде зупинка роботи бази даних командою: *sudo service mysql stop*

На інших вузлах кластера СКБД також слід встановити відповідне програмне забезпечення та зупинити роботу сервісу *MySQL* для коректного настроювання спільної роботи вузлів. Параметри настройки вузла слід вказати у конфігураційному файлі, наприклад, у *Ubuntu Server* виконаємо редагування параметрів за допомогою текстового редактору *nano*:

sudo nano /etc/mysql/my.cnf

Для збереження даних у редакторі *nano* слід застосовувати скорочення: "*Ctrl+O*", а для виходу з редактору – "*Ctrl+X*".

Фактично у конфігураційному файлі слід визначити загальне ім'я кластеру та вказати IP-адреси відповідних вузлів, з яких буде складатися кластер на базі СКБД *Percona XtraDB Cluster.*

Відповідно до обраної топології локальної мережі кластеру для його першого вузла слід додати до конфігураційного файлу (*my.cnf*) наступні параметри:

[*mysqld*]
Шлях до бібліотеки *Galera*
wsrep_provider=/usr/lib/libgalera_smm.so

Загальна назва кластеру та IP-адреси його вузлів
wsrep_cluster_name=lab-cluster
wsrep_cluster_address=gcomm://192.168.0.101,192.168.0.103,
192.168.0.105

Ім'я вузла та його IP-адреса
wsrep_node_name=node1
wsrep_node_address=192.168.0.101

Застосування *XtraBackup* для узгодження даних між вузлами
SST – *State Snapshot Transfer*
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth=sstuser:passw0rd

Не застосовуємо експериментальні можливості та функціонал,
який не підтримуються *Percona XtraDB Cluster*
pxc_strict_mode=ENFORCING

Формат реплікації та система збереження таблиць
за матеріалами https://www.percona.com/doc/
percona-xtradb-cluster/5.7/configure.html
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2

Для інших вузлів кластеру слід також виконати відповідні зміни у конфігураційному файлі сервісу *MySQL*. Однак, у параметрах, які специфічні для вузла, слід вказати вірні значення для "*wsrep_node_name*" та "*wsrep_node_address*". Остаточне розгортання кластеру можна виконати автоматично за командою, яку слід виконати на першому вузлі з правами доступу суперадміністратора: *sudo su*

/etc/init.d/mysql bootstrap-pxc

Потім для додавання інших вузлів варто буде тільки запустити відповідні сервіси на вузлах: */etc/init.d/mysql start*

До того як активувати режим автоматичного формування кластеру слід на першому вузлі (тобто на якому запускаємо скрипт *bootstrap-pxc*) слід додати користувача із дозволом на реплікацію бази даних:

mysql -u root -p

> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 'passw0rd';

*> GRANT RELOAD, LOCK TABLES, PROCESS, REPLICATION CLIENT ON *.* TO 'sstuser'@'localhost'*;

> FLUSH PRIVILEGES;

Після розгортання *Percona XtraDB Cluster* можна визначити поточні параметри роботи кластеру (рис. 10):

> show status like 'wsrep%';

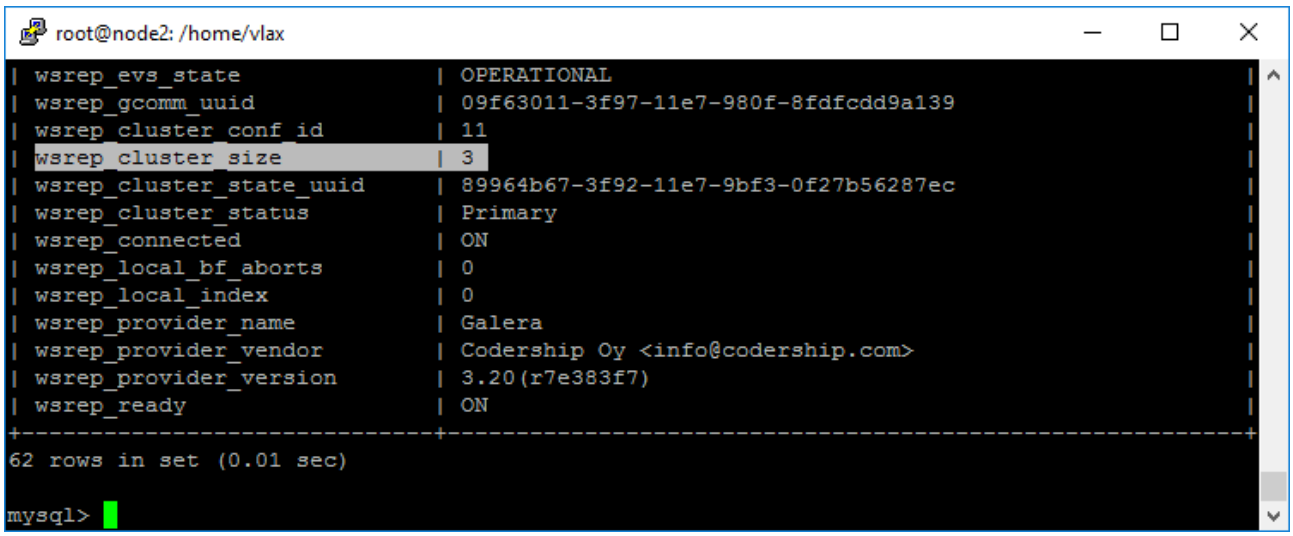

Рис. 10. **Поточний стан *Percona XtraDB Cluster***

2. Виконати тестування кластера у разі виходу з ладу його вузлів.

3. Скласти звіт із лабораторної роботи.

Контрольні питання:

1. Наведіть приклад типової архітектури WEB-сервісу, що має високе навантаження.

2. Наведіть приклади та поясніть особливості застосування технології кластерних СКБД.

3. Наведіть типову структуру кластера баз даних. Які типи вузлів використовуються у наведеній архітектурі?

4. Назвіть мінімальну кількість вузлів, яких буде достатньо для формування кластера *Percona XtraDB*.

5. Назвіть з яких складових буде формуватися сукупна вартість підтримки та супроводження певного рішення розподіленого сховища даних.

Лабораторна робота 5
Розгортання та налаштування кластера на базі СКБД *PostgreSQL* із застосуванням віртуальних машин

Мета лабораторної роботи

Оволодіти навичками адміністрування реляційної бази даних *PostgreSQL*. Визначити засоби масштабування, забезпечення безвідмовної роботи та доступності СКБД *PostgreSQL*. Ознайомлення з кластерними технологіями на основі *Postgres-XL*.

Загальні тези лабораторної роботи

Швидкість роботи, взагалі не є основною причиною використання СКБД. Більш того, перші реляційні бази працювали повільніше своїх попередників. Вибір цієї технології був викликаний скоріше:

можливістю покласти підтримку цілісності даних на СКБД;

незалежністю логічної структури даних від фізичної.

Ці особливості дозволяють сильно спростити написання програм, але вимагають для своєї реалізації додаткових ресурсів. Таким чином, перш ніж шукати відповідь на питання "як змусити СКБД працювати швидше для певної задачі?", слід відповісти на питання "чи немає ефективнішого засобу для вирішення завдання прискорення швидкодії, наприклад, застосувати розподілену СКБД?". Таким чином, застосування розподілених систем може значно вплинути на продуктивність інформаційної системи чи комплексу.

PostgreSQL – це об'єктно-реляційна система управління базами даних, заснована на *POSTGRES*, яка була розроблена на факультеті комп'ютерних наук Каліфорнійського університету в Берклі у середині 80-х. У *POSTGRES* з'явилась велика кількість нововведень, які були реалізовані в деяких комерційних СКБД вже набагато пізніше. *PostgreSQL* – СКБД із відкритим вихідним кодом. Вона підтримує більшу частину стандарту SQL і пропонує багато сучасних функцій:

складні запити;

зовнішні ключі;

тригери;

змінювані уявлення;

транзакційна цілісність;
багатоверсійність.

Завдяки вільній ліцензії, *PostgreSQL* дозволяється безкоштовно використовувати, змінювати і поширювати всім і для будь-яких цілей – особистих, комерційних чи навчальних.

Користувачі можуть розширювати можливості *PostgreSQL*, наприклад, створюючи свої:

типи даних;
функції;
оператори;
агрегатні функції;
методи індексування;
процедурні мови.

Postgres-XL – засіб, який дозволяє об'єднати кілька кластерів *PostgreSQL* таким чином, щоб вони працювали як один вузол бази даних. Для клієнта, який підключається до такого кластера, немає ніякої різниці, чи працює він із єдиним вузлом *PostgreSQL* або з кластером *Postgres-XL*. Система *Postgres-XL* пропонує декілька режимів розподілу таблиць у кластері: реплікація та шардінг. При реплікації всі вузли містять однакову копію таблиці, а для шардінгу (або сегментування) дані рівномірно розподіляються серед членів кластера. Поточна реалізація заснована на *PostgreSQL*.

Postgres-XL складається з трьох типів компонентів (рис. 11): глобальний монітор транзакцій (Global Transaction Manager), координатори (Coordinators) і вузли даних (*DataNodes*).

GTM (Global Transaction Manager) – відповідає за забезпечення вимог ACID (Atomicity, Consistency, Isolation, Durability – набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність). Застосування окремого сервера для GTM та його резервування є оптимальним рішенням, а для об'єднання множинних запитів і відповідей від координаторів і вузлів даних запущених на одному сервері має сенс налаштувати GTM-проксі. Завдяки цьому можна знизити навантаження на GTM та зменшити загальну кількість взаємодій з ним.

Координатор – центральна частина кластера. Саме з ним взаємодіє клієнтська програма. Управляє призначеними для користувача сесіями і взаємодіє з GTM і вузлами даних. Фактично координатор виконує аналіз запитів, будує план їх виконання та відсилає запити на кожен з компонентів кластера, потім збирає результати і відсилає їх назад клієнту.

Координатор не зберігає ніяких призначених для користувача даних. Він зберігає тільки службові дані, щоб визначити як обробляти запити, де знаходяться вузли даних. У разі виходу з ладу одного з координаторів можна просто переключитися на інший.

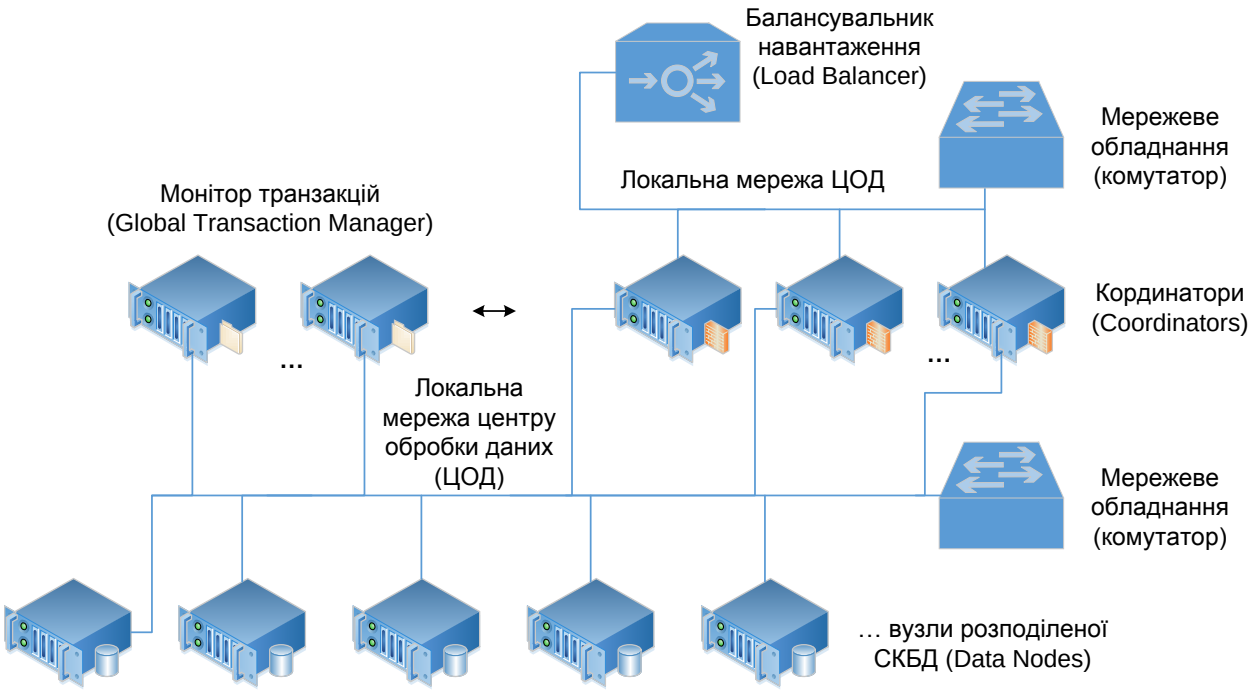

Рис. 11. **Спрощена архітектура кластера *Postgres-XL***

Вузол даних – місце де зберігаються призначені для користувача дані та індекси. Зв'язок з вузлами даних здійснюється тільки через координатори. Для забезпечення високої доступності можна застосувати резервування копій вузлів. Для рішень завдань балансувальника навантаження у кластері можна використовувати технології pgpool-II [18; 23; 35; 38; 39; 45; 59].

Завдання до лабораторної роботи:

1. У середовищі віртуалізації виконати розгортання *PostgreSQL*.
2. Ознайомитися із основами роботи *PostgreSQL* та особливостями масштабування цієї СКБД. Розгорнути екземпляр вузла PostgreSQL.
3. Ознайомитися із основами розгортання *Postgres-XL* за матеріалами офіційної документації (http://www.postgres-xl.org/).
4. Скласти звіт із лабораторної роботи.

Контрольні питання:

1. Наведіть визначення реляційної бази даних. Наведіть основні сучасні СКБД, що застосовують реляційну модель збереження даних та виконайте їх порівняння.

2. Наведіть приклади та поясніть особливості застосування технології кластерної СКБД *Postgres-XL*.

3. Поясніть значення закону Амдала у рішенні завдань побудови кластерних рішень? Чи можливо застосовувати відповідний закон у разі формування кластерної бази даних?

4. Назвіть обмеження, які визначає теорема Брюера (CAP) для побудови розподілених сховищ даних.

5. Назвіть відмінні особливості стратегії застосування розподілених сховищ даних на основі кластерної СУБД *Postgres-XL*. Для рішення яких бізнес-завдань доцільно застосовувати рішення рівня *Postgres-XL*.

Лабораторна робота 6
Розгортання та робота із розподіленою базою даних на основі СКБД *MongoDB* у середовищі віртуальних машин

Мета лабораторної роботи

Оволодіти навичками адміністрування бази даних *MongoDB*. Визначити засоби масштабування, забезпечення безвідмовної роботи та доступності нереляційної СКБД. Програмування рішень завдань оброблення даних за технологією *Map Reduce* та застосування *Aggregation framework*.

Рекомендації щодо підготовки до виконання лабораторної роботи

Для виконання лабораторної роботи слід ознайомитися із основними властивостями *MongoDB* – документ-орієнтованої системи керування базами даних з відкритим вихідним кодом, яка забезпечує високу продуктивність, високу доступність та автоматичне масштабування [78]. Запис у системі *MongoDB* є документом, який має структуру даних. Фактично запис у нереляційній базі даних, якою є *MongoDB*, складається із записів пар значень, що схожі на об'єкти JSON (*JavaScript Object Notation*). Значення окремих записів можуть містити інші документи, масиви та масиви документів (рис. 12).

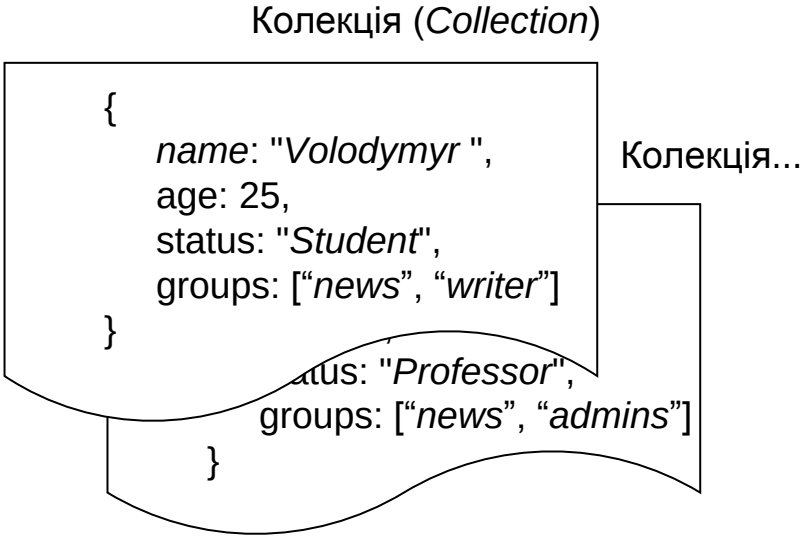

Рис. 12. **Структура даних *MongoDB***

Записи групуються у колекції. Фізично дані зберігаються у форматі BSON, який є бінарним сховищем даних типу JSON.

Перевагами використання документів є:

документи (або об'єкти) відповідають власним типам даних у багатьох мовах програмування;

вбудовані документи і масиви зменшують необхідність у складних запитах та з'єднань даних типу *JOIN*;

завдяки динаміці схеми бази даних, нереляційні рішення підтримують об'єктно-орієнтований механізм поліморфізму.

MongoDB підтримує зручну мову запитів для виконання операцій читання і запису CRUD (операції створення – "*Create*", зчитування –"*Read*", оновлення – "*Update*" та видалення – "*Delete*") й також має зручний інтерфейс командного рядка, що застосовує засоби мови *JavaScript* для роботи з даними. Для вибору бази даних застосовується команда:

use myNewDB

Створити колекцію можна безпосередньо виконав запит, наприклад, додав запис:

```
db.myNewCollection1.insert ( {    name: "Volodymyr ",
                                  age: 25,
                                  status: "Student" } )
```

Зчитування даних виконується командою, наприклад, пошуку:

```
db.myNewCollection1.find ( {      age: { $gt: 18 } },
                           {      name: 1 } ). limit(5)
```

Оновлення документа можна виконати командою, наприклад, що має наступний вигляд:

```
db.collection.update(
  <query>,
  <update>,
  {
    upsert: <boolean>,
    multi: <boolean>,
    writeConcern: <document>
  }
)
```

де *collection* – ім'я колекції; *<query>* – критерії пошуку даних, *<update>* – дані для оновлення; *upsert* – параметр, що вказує на необхідність створення запису з новими даними, навіть у разі коли запис, за яким виконується пошук був відсутній (за замовчуванням цей параметр має значення – *false*); *multi* – дозволяє виконати заміну даних за всіма записами, що задовольняють критерію пошуку (за замовчуванням цей параметр має значення – *false*, що забезпечую заміну тільки першого запису, який було знайдено у колекції); *writeConcern* – застосування механізму забезпечення надійності роботи бази даних, частіше використовується в завданнях, що базуються на масштабуванні системи.

Видалити всі документи колекції можна за допомогою команди, наприклад:

```
db. myNewCollection1.remove( { } )
```

Цей метод очищення колекції більш ефективний у разі рішення завдань повторного створення колекції, на відміну від застосування методу *drop*(), який видаляє колекцію поруч із видаленням індексів.

Слід відзначити, що база даних *MongoDB* забезпечує механізми агрегування даних, як на рівні традиційних запитів *mapReduce* так й спеціалізованого механізму *Aggregation Framework*, що дозволяє ефективно застосовувати відповідні рішення для вирішення завдань обробки великих даних [36; 37; 62]. Також до особливостей *MongoDB* можна віднести реалізацію пошуку за текстом та забезпечення геопросторових запитів. База даних має високу доступність, засоби реплікації та масштабування.

Загальні тези лабораторної роботи

База даних *MongoDB* дозволяє ефективно вирішувати завдання масштабування даних. Вона підтримує горизонтальне і вертикальне

масштабування. Звичайно вертикальне масштабування забезпечується завдяки нарощування потужності серверу за допомогою додавання процесорних модулів чи ядер, збільшення обсягу оперативної пам'яті та збільшення дискового простору чи застосування швидших типів носіїв. Горизонтальне масштабування забезпечується додаванням нових екземплярів серверів керування базою даних машин до існуючих і розподіл даних між ними [64; 65]. Для вертикального масштабування схема розгортання *MongoDB* не відрізняється від звичайного встановлення на вузол (рис. 13).

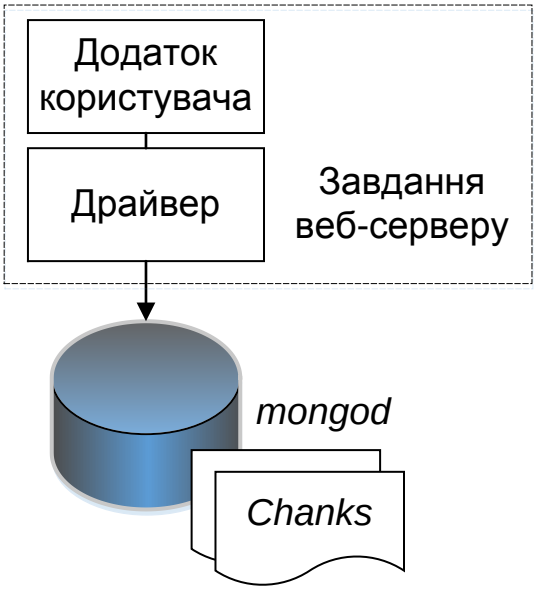

Рис. 13. **Установка *MongoDB* на окремий сервер (вузол)**

Для доступу до бази даних додаток користувача за допомогою драйверу або за допомогою інтерфейсу керування командного рядка (*Shell*) взаємодіють із процесом *mongod*. Фактично *mongod* – це основний процес, що є системою керування базою даних, який призначений для отримання, оброблення та виконання запитів користувача. Дані в *mongod* зберігаються в *chunks*. Кожен *chunk* має розмір, який за замовчуванням складає 64 Мб. Фізично *chunks* зберігаються в файлах "*dbName.n*", де *n* – номер за порядком. У разі досягнення розміру 64 Мб (або іншого розміру, що може встановити адміністратор бази даних) *chunk* ділиться навпіл і наступний файл займає розмір у двічі більший, наприклад, 128 Мб, 256 Мб та далі відповідно до зростання даних. Після такого сегментування *chunk* може збільшуватися максимум до 2 Гб, після чого зростання розміру припиняється і *mongod* починає створювати файли одного й того ж розміру. Тому навіть у разі відносно невеликого розміру реальних даних, можна виявити, що система займає багато простору на жорсткому диску.

У разі рішення завдання масштабування доцільно залучити механізм шардінгу (*Sharding*) бази даних *MongoDB* (рис. 14).

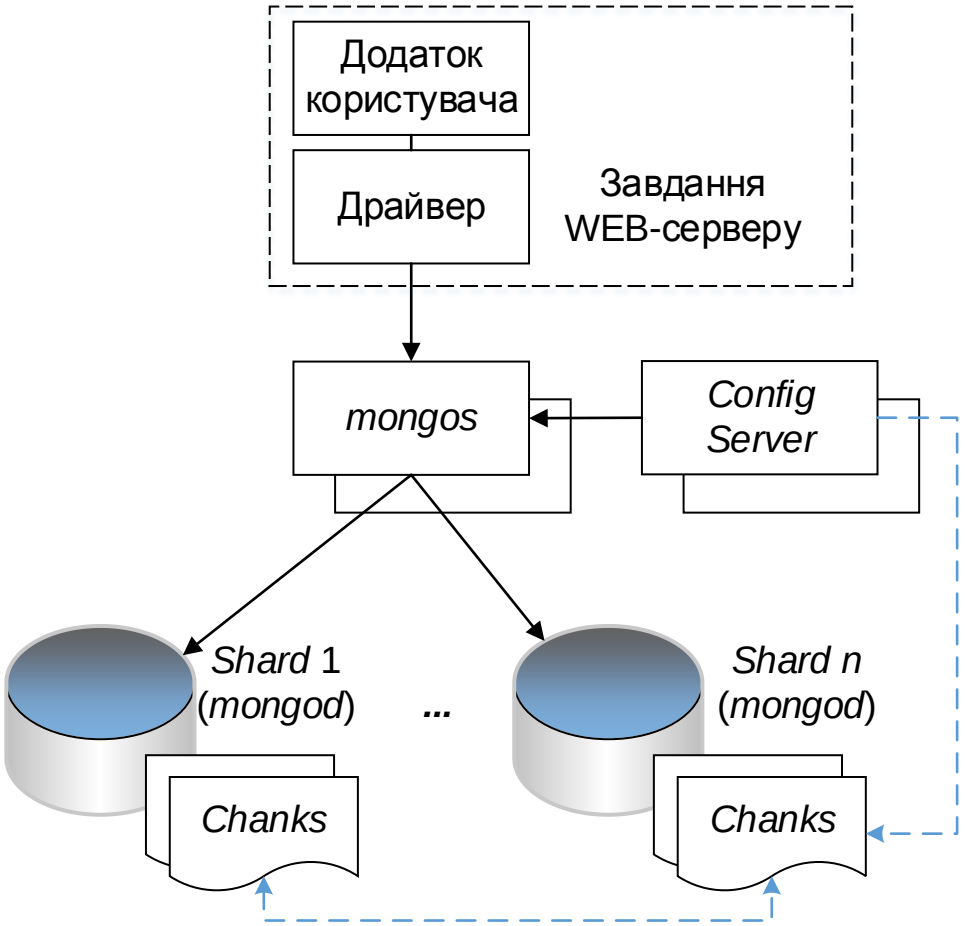

Рис. 14. **Спрощена структура масштабування бази даних MongoDB**

Шард або сегмент – це екземпляри *mongod*, які розповсюджуються по вузлах кластера. Для використання в умовах виробництва (*production*), кожен шард повинен мати репліки (*replicaSet*) для надійного збереження даних (що не виключає виконання дій адміністратора щодо резервного копіювання вмісту бази даних).

Сервер конфігурацій (*config*) не виконує жодних клієнтських запитів, а тільки зберігає метадані кластера та відповідає за організацію розміщення даних за вузлами системи.

Сервер маршрутизації є екземпляром *mongos* – фактично виконує завдання маршрутизації запитів та, відповідно, виконує балансування навантаження на вузли кластера.

Особливістю застосування шардінгу є те, що у разі масштабування дані не просто записуються в *chunks*, а потрапляють у них за певним

діапазоном визначеного поля – *Shard Key*. Система керування базою даних сама обирає оптимальний критерій сегментування даних за *chunks*, однак, адміністратор бази даних може самостійно обрати ключ, за яким буде виконано відповідний розподіл даних. Тому вибір *Shard Key* повинен бути спрямований на сприяння рівномірного розподілу даних між *chunks.*

Слід відзначити, що теоретично кластер *MongoDB* має властивості нескінченного масштабування, за умови додавання нових вузлів до кластеру, та забезпечує якісну безпеку даних, за умови використання механізму реплікації. Наприклад, вузли реплік можуть розташовуватися в різних центрах обробки даних, навіть у значно георозподілених зонах.

Завдання до лабораторної роботи:

1. У середовищі віртуалізації виконати базову установку операційної системи *Ubuntu Server 14.04 LTS*. Виконання розгортання серверної операційної системи слід виконувати відповідно до рекомендацій розділу *Installation* офіційного керівництва *Ubuntu 14.04 Server Guide* (https:// help.ubuntu.com/lts/serverguide /installing-from-cd.html).

Параметри віртуальної машини мають бути такими або кращими за: однин процесор, 1 Гб – оперативної пам'яті, 8 Гб – накопичувач (віртуальний диск). Мережевий інтерфейс зручніше конфігурувати як *Bridged Networking* (для проведення лабораторної роботи можна застосувати DHCP-сервіс навчальної мережі, однак слід зазначити, що у промислових умовах кожен сервер повинен мати відокремлену фіксовану IP-адресу).

Кожна віртуальна машина повинна містити базову установку серверної операційної системи, що не містить додаткових сервісів, крім *OpenSSH server* для віддаленого доступу до віртуальної машини.

2. Виконати розгортання *MongoDB* у середовищі віртуальних машин відповідно до *Install MongoDB Community Edition on Ubuntu* керівництва *The MongoDB* 3.2 *Manual* (https://docs.mongodb.com/manual/tutorial/ install-mongodb-on-ubuntu/).

3. На основі офіційного керівництва *The MongoDB* 3.2 *Manual* виконати дії щодо створення бази даних, колекції. На цій основі вивчити особливості програмування запитів типу *mapReduce* та застосування спеціалізованого механізму *Aggregation Framework*.

4. Виконати масштабування бази даних на два чи більше вузла на основі застосування механізму Sharding.

5. Забезпечити високу надійність рішення на базі *MongoDB* на основі механізму реплікації.

6. Скласти звіт із лабораторної роботи.

Контрольні питання:

1. Що таке великі дані (*Big Data*)? У чому є особливості застосування розподілених сховищ даних у рішенні задач обробки великих даних?

2. Наведіть визначення нереляційної бази даних. Що таке реплікація та сегментування бази даних? Наведіть приклади застосування.

3. Назвіть основні характеристики алгоритму *MapReduce*.

4. Назвіть особливості розробки програмного забезпечення, що застосовує технології розподілених сховищ даних.

5. Назвіть відмінні особливості стратегії застосування розподілених сховищ даних для рішення бізнес-завдань розвинутого сервісу соціальної мережі.

Лабораторна робота 7
Розгортання та адміністрування системи моніторингу Nagios у середовищі віртуальних машин

Мета лабораторної роботи

Визначити особливості застосування сучасних систем моніторингу стану вузлів та сервісів у GRID-системах та WEB-рішеннях. Оволодіння навичками розгортання та налаштування систем моніторингу *Nagios*, *Icinga*.

Загальні тези лабораторної роботи

Розглянемо застосування відкритих систем для оцінки стану локальної мережі *RRDtool* (http://oss.oetiker.ch/rrdtool/index.en.html) – набір утиліт для роботи з кільцевими базами даних (*Round-Robin Database*). Такі бази даних застосовуються для зберігання й обробки динамічних (змінних у часі) послідовностей даних, таких як мережевий трафік, пропускна здатність мережі, температура та завантаження процесору. Усі дані зберігаються у кільцевій базі, розмір якої залишається незмінним (за матеріалами з Вікіпедії).

RRDtools, крім іншого, містять у собі можливість графічного відображення інформації, що збережена. Даний набір утиліт поширюється за ліцензією GNU GPL. Відмінною рисою технології кільцевих баз даних (RRD) є робота з фіксованою кількістю даних і покажчиком на поточний елемент. Таку базу даних можна подати у вигляді кільця із центром, що вказує на конкретну крапку, в якої дані можуть бути збережені або прочитані. Коли поточні дані прочитані або записані, покажчик переміщується в наступний елемент. Оскільки така база даних є кільцевою, тобто без початку та кінця, вона не може переповнитися. У такій базі дані записуються вперед і вперед, але через якийсь час, усі доступні місця будуть використані та процес автоматично почне використати старі місця для запису нових даних.

Таким чином, така база даних надає можливості зберігати дані протягом тривалого часу. Наприклад, можна реєструвати події через визначений інтервал часу (приблизно п'яти хвилин) та заносити їх в базу даних, що може накопичувати дані за час. На основі цієї бази вести базу за день, іншу – за місяць, так далі. Але недоліком визначеної технології є відсутність даних про стан системи, наприклад, за два з половиною часу раніше, так як вони стають однією точкою у даних за день. Однак велика деталізація у процесі вирішення задач моніторингу мережі рідко коли буває потрібна, тому технологія *RRDtool* набула великого поширення.

Для роботи з системою за технологією *RRDtool* системний адміністратор повинен розробити програмні засоби, що будуть виконувати реєстрацію даних та передавати їх до бази даних, яка у свою чергу засобами *RRDtool* буде відображуватися. Однак є багато проектів, що базуються на *RRDtool* та надають вже готові рішення для збору статистики про стан одиниць обчислювальної мережі. Наприклад, відкрита система *Cacti* (http://www.cacti.net/).

Налаштувати *Cacti* на вузлу можна за рекомендаціями How to Install and Setup Cacti on Ubuntu 16.04 Інтернет-видання *LinOxide* (https://linoxide.com/ ubuntu-how-to/ install-setup-cacti-ubuntu-16-04/). Спочатку слід оновити операційну систему:

 # apt-get update

 # apt-get upgrade

Потім слід додати репозиторій *Cacti* та розгорнути стек LAMP:

 # add-apt-repository 'deb http://archive.ubuntu.com/ubuntu
trusty universe'

 # apt-get update

 # apt-get install apache2 mysql-server-5.6 php libapache2-mod-php

Для роботи *Cacti* слід встановити утиліти SNMP, сервіс "snmpd" та *RRDtool*, що дозволить виконувати моніторинг вузла "localhost":

 # apt-get install snmp snmpd rrdtool

apt-get install cacti cacti-spine

/etc/init.d/mysql start

У процесі встановлення *Cacti* слід вказати, що буде застосовано WEB-сервер *apache2*, а також визначити або згенерувати автоматично пароль на доступ до бази даних сервісу *Cacti.* Подальша установка системи моніторингу виконується у середовищі браузера за адресою, яка відповідає віртуальному вузлу, наприклад, http://192.168.0.107/cfcti/. У WEB-інтерфейсі установки фактично слід тільки визначити пароль користувача системи "admin".

На рис. 15 наведено зовнішній вигляд екрану у робочий момент роботи системи *Cacti*. Можна обрати два режиму роботи у WEB-додатку: Console – для конфігурування системи, додавання інших вузлів для моніторингу їх стану, та Graphics – графічне відбиття параметрів роботи сервісів та вузлів, що відстежуються.

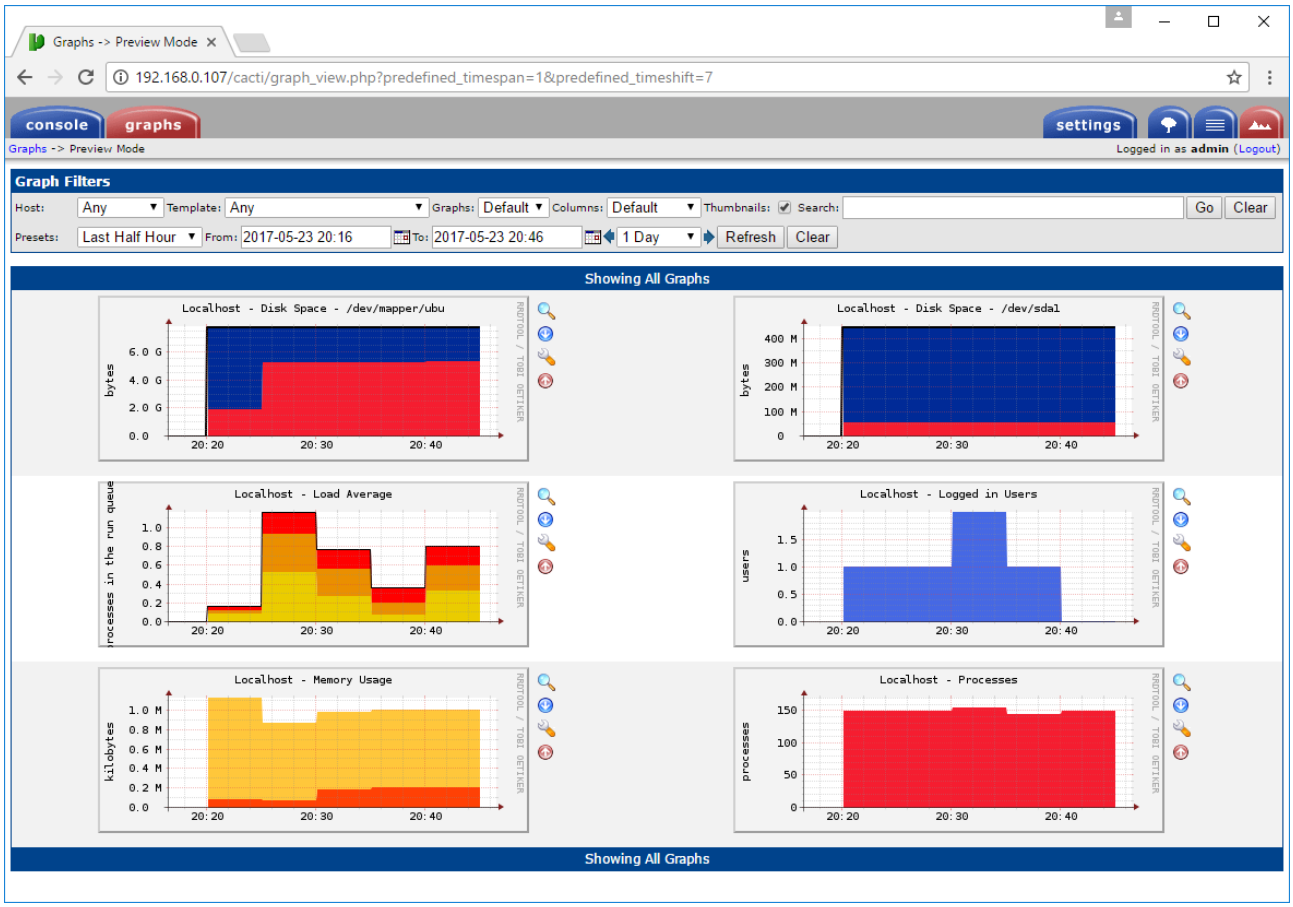

Рис. 15. **Приклад роботи системи *Cacti***

Cacti дозволяє знімати облікові дані з будь-якого мережевого пристрою за протоколом SNMP (за умови, що така апаратура працює з цим протоколом). Однією з переваг *Cacti* порівняно з іншими більш простими системами типу MRTG, можна відзначити зручний WEB-інтерфейс, розширюваність за рахунок написання простих модулів для реалізації необхідних додатковий функцій, наявність готових шаблонів для різного мережевого встаткування. Однак, установка та налагодження роботи *Cacti* потребує від системного адміністратора більше часу порівняно з MRTG, також для своєї роботи *Cacti* необхідна наявність бази даних *MySQL*, підтримки Web-сервером мови PHP та системи *RRDtool* на сервері де він буде встановлений. Таким чином, *Cacti* дозволяє накопичувати дані про стан мережевого встаткування, яке підтримує протокол SNMP, наприклад: комутатори, що керуються; маршрутизатори; робочі станції та сервера під керуванням операційних систем *Unix, Linux, Windows* чи ін. (за матеріалами http://wiki.asplinuxclub.org).

Функціональнішою системою моніторингу стану комп'ютерної мережі є система *Nagios* (http://www.nagios.org/). Ця програма з відкритим кодом, стежить за визначеними вузлами й службами, і сповіщає адміністратора в тому випадку, якщо якісь зі служб припиняють (або відновлюють) свою роботу. За допомогою цього програмного комплексу можна виконувати моніторинг мережевих служб (SMTP, POP3, HTTP, NNTP, ICMP, SNMP), моніторинг стану комп'ютерів (завантаження процесора, використання диска, системні повідомлення). Ця система підтримує дистанційний моніторинг через шифровані тунелі SSH або SSL.

Проста архітектура модулів розширень *Nagios* дозволяє, використовуючи практично будь-яку мову програмування, легко розробляти свої власні способи перевірки служб [16; 17; 24; 30; 33; 34; 57]. У системі, що розглядається можна визначати ієрархію мережевих засобів і розрізняти ланки які вийшли з ладу від тих, що, недоступні (рис. 16).

Nagios передбачає відправлення повідомлень у випадку виникнення проблем зі службою або пристроєм у мережі (наприклад, за допомогою електронної пошти). Також у системі передбачена можливість організації спільної роботи декількох систем моніторингу з метою підвищення надійності (за матеріалами з Вікіпедії).

Налаштувати сервер *Nagios* на вузлі можна за рекомендаціями *How to Install Nagios Server Monitoring on Ubuntu* 16.04 Інтернет-видання

Howtoforge (https://www.howtoforge.com/tutorial/ubuntu-nagios/). Спочатку слід встановити залежності системи:

sudo apt-get install wget build-essential apache2 php apache2-mod-php7.0 php-gd libgd-dev sendmail unzip

Потім слід додати нового користувача та групу для розгортання *Nagios*:

 # useradd nagios

 # groupadd nagcmd

 # usermod -a -G nagcmd nagios

 # usermod -a -G nagios,nagcmd www-data

Виконати розгортання серверу моніторингу із вихідних кодів:

 # cd ~

 # wget https://assets.nagios.com/downloads/nagioscore/releases/ nagios-4.2.0.tar.gz

 # tar -xzf nagios.tar.gz*

 # cd nagios-4.2.0

Виконати компіляцію та установку *Nagios*:

./configure --with-nagios-group=nagios --with-command-group=nagcmd

 # make all

 # make install

 # make install-commandmode

 # make install-init

 # make install-config

 # /usr/bin/install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-available/nagios.conf

 # cp -R contrib/eventhandlers/ /usr/local/nagios/libexec/

 # chown -R nagios:nagios /usr/local/nagios/libexec/eventhandlers

Також доцільно додати плагіни *Nagios*, які теж слід скомпілювати:

 # cd ~

 # wget https://nagios-plugins.org/download/nagios-plugins-2.1.2.tar.gz

 # tar -xzf nagios-plugins.tar.gz*

 # cd nagios-plugins-2.1.2

./configure --with-nagios-user=nagios --with-nagios-group=nagios --with-openssl

 # make

 # make install

Після компіляції та установки системи моніторингу слід налаштувати параметри його роботи: *nano /usr/local/nagios/etc/nagios.cfg*

У файлі конфігурацій варто розкоментувати рядок:

cfg_dir=/usr/local/nagios/etc/servers

Відповідно до попередніх дій щодо розгортання системи *Nagios*, слід додати файл із описом топології мережі, для якої буде виконуватися завдання моніторингу: *mkdir -p /usr/local/nagios/etc/servers*

nano /usr/local/nagios/etc/servers/ubuntu_host.cfg

Файл конфігурації мережі відбиває основні настроювання щодо опису параметрів всіх вузлів і містить визначення відповідних команд моніторингу, як вузлів, так й сервісів на певних вузлах:

```
# Параметри моніторингу вузла
define host {
    use             linux-server
    host_name       ubuntu_host
    alias           Ubuntu Host
    address         192.168.0.101    }
# Основні параметри моніторингу сервіса
define service {
    host_name               ubuntu_host
    service_description     PING
    check_command           check_ping!100.0,20%!500.0,60%
    max_check_attempts      2
    check_period            24x7        }
```

На вузлах, для яких виконуються завдання моніторингу слід встановити віддалений агент *NRPE Service*:

sudo apt-get install nagios-nrpe-server nagios-plugins

У файлі конфігурації */etc/nagios/nrpe.cfg* слід вказати сервер *Nagios* із якого дозволяється отримувати статистику роботи визначеного вузла.

Відзначимо, що для розгортання системи моніторингу, що розглядається, також слід буде налаштувати авторизацію на web-сервері *Apache*, встановити *Nagios* як сервіс та ін.

Наприклад, для виконання послідовності налаштування *Nagios Server* можна скористатися рекомендаціями *How To Install Nagios 4 and Monitor Your Servers on Ubuntu 14.04* користувачів сервісу *DigitalOcean* (https:// www.digitalocean.com/ community/ tutorials/ how-to-install- nagios-4-and-monitor-your-servers-on-ubuntu-14-04).

Таким чином, найкращим рішенням для організації системи моніторингу стану розподіленого сховища даних може стати організація виділеного

серверу (рис. 16). Така система повинна відбивати стан компонентів мережі та сповіщати адміністратора про нештатні ситуації.

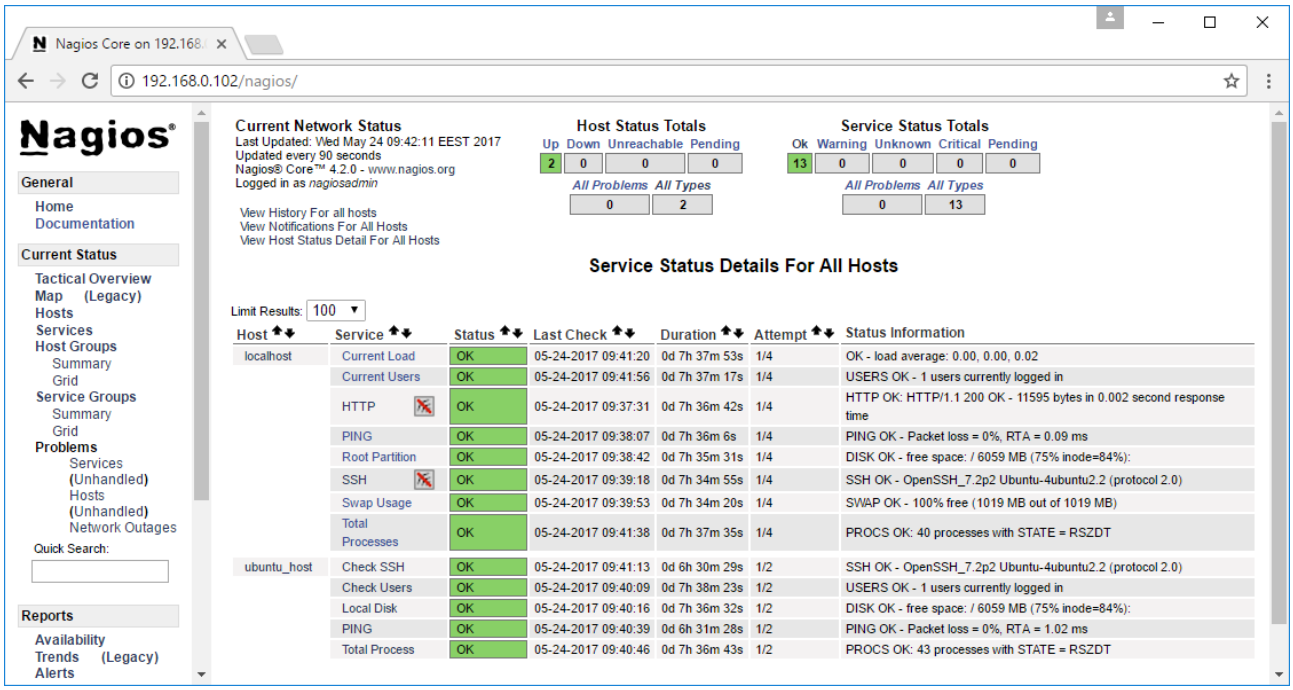

Рис. 16. **Приклад роботи системи *Nagios***

Слід відмітити, що зараз існує багато відкритих систем для моніторингу стану обчислювальної мережі, наприклад, *ZABBIX* (http:// www.zabbix.com/), яка застосовує реляційну базу даних для збереження даних моніторингу вузлів та сервісів, на відміну від технологій *RRDtool.* Цікавими є рішення застосування системи моніторингу *Icinga* (https://www.icinga.com/) та ін.

Також існують складні масштабовані рішення, наприклад, що базуються на поєднані декількох систем (за матеріалами конференції *ThinkPHP* #14): відбиття даних – Grafana (https://grafana.com/), спеціалізованої бази даних – Influx (https://www.influxdata.com/) та агента – Telegraf (https:// github.com/ influxdata/telegraf). Поруч із цим, для проектів різного масштабу та складності можна застосувати системи моніторингу, які надаються як сервіс хмарних обчислень, наприклад, рішення *New Relic* (https://newrelic.com/) та ін.

Завдання до лабораторної роботи:

1. У середовищі віртуалізації виконати розгортання систем *Cacti* та *Nagios*. Порівняйте їх функціональні можливості, інтерфейс та зручність розгортання й практичного застосування.

2. Додати декілька віртуальних вузлів та сконфігурувати систему моніторингу щодо відстеження стану цих вузлів.

3. Змоделювати стан виходу з ладу вузла, моніторинг якого виконує система *Nagios* та знову повернути його до працездатного стану. Проаналізувати відбиття відповідної інформації системою моніторингу.

4. Скласти звіт із лабораторної роботи.

Контрольні питання:

1. Що таке хмарні технології? Назвіть особливості застосування розподілених сховищ даних для рішення задач *Cloud Computing*.

2. Назвіть основні системи моніторингу стану розподілених систем. Вкажіть їх особливості та основні приклади застосування.

3. Чим відрізняється SQL-модель баз даних від *NoSQL* рішень? Поясніть, чи є різниця у технічних засобах моніторинга SQL баз даних та рішень типу *NoSQL?*

4. Назвіть основні напрями масштабування рішень моніторингу на базі технології *Nagios*.

5. Назвіть відмінні особливості стратегії застосування розподілених сховищ даних для рішення бізнес-завдань великого Інтернет-магазину. Вкажіть місце засобів моніторингу стану IT-ресурсів у такому рішенні, наведіть приклади.

Додатки

Додаток А

Розгортання операційної системи *CentOS* у середовищі віртуалізації

Розглянемо приклад установки серверного програмного забезпечення. У якості операційної системи обрано дистрибутив *CentOS* 7 (http://www. centos.org/). Цей дистрибутив GNU/Linux, засновано на комерційному *Red Hat Enterprise Linux* (https://access.redhat.com/ documentation /ru-RU /Red_Hat_Enterprise_Linux /7 /html /Installation _Guide /index.html), що позиціонується для корпоративного використання. Основна особливість дистрибутива – висока надійність та безпека, наявність комерційної підтримки. Важливим є те, що багато виробників програмного та апаратного забезпечення включили RHEL у число дистрибутивів GNU/*Linux*, що мають їх підтримку.

Слід відзначити що, *Red Hat Enterprise Linux* складається з вільного програмного забезпечення з відкритим кодом, який є доступним у вигляді дисків із бінарними пакетами тільки для платних передплатників. Згідно з умовами ліцензії GPL та ін., *Red Hat* надає всі вихідні коди, а розроблювачі *CentOS* (*Community Enterprise Operating System*) використовують цей вихідний код для створення продукту, дуже близького до *Red Hat Enterprise Linux* (за матеріалами з Вікіпедії).

Розгортання сервера на базі *CentOS* 7 надає можливості застосувати багато сервісів, що є типовими, наприклад, для провайдерів послуг *Internet* (ISP) та хостингу: WEB-сервер *Apache* (з підтримкою захищеної передачі даних SSL), поштовий сервер *Postfix* із авторизацією SMTP та сервер *Dovecot* із авторизацією POP3/IMAP, DNS-сервер BIND, файловий сервер *Proftpd*, база даних *MySQL*, система контролю за розміром обсягу дискового простору для користувача (*Quota*), мережевий екран (*Firewall*), та ін. (http:// howtoforge.com/ perfect-server-centos-5.2).

Для того щоб почати установку дистрибутива CentOS 7 треба завантажити образ установочного DVD-носія із WEB-сайту розробників (http:/ isoredirect.centos.org/centos/7/isos/x86_64/CentOS-7-x86_64-DVD-1611.iso). Оскільки для сучасних серверних систем слід надавати перевагу 64-бітній архітектурі для програмного забезпечення операційної системи, оберемо для розгортання дистрибутив *CentOS*, який є доступним для платформи x86_64.

Вибір 64-бітної архітектури пов'язано з тим, що фактично всі розробники серверних операційних систем переважно розробляють тільки продукти для x86_64, архітектури ARM-сумісних систем та незначною кількістю інших платформ.

Зазначимо, що установка для 32-х та 64-х розрядних мікропроцесорних архітектур є практично однаковою та починається з перевірки якості носія (рис. А.1). Потім слід обрати мову та перейти до подальшого конфігурування компонентів для установки у графічному режимі.

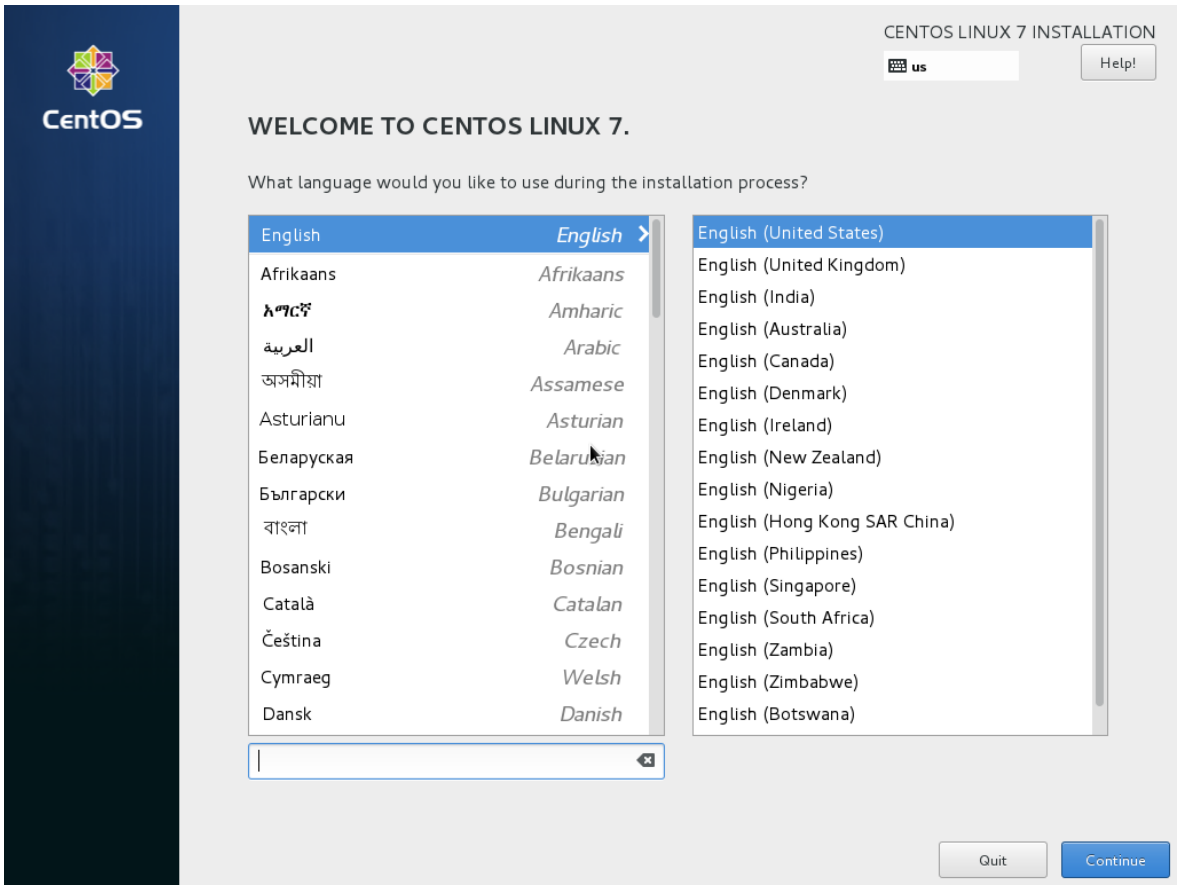

Рис. А.1. **Установка *CentOS* 7. Вибір мови дистрибутиву**

Виконання перевірки установочного комплекту дисків є важливим, тому що якщо не виконати цю операцію на початку установки, то потім можливі серйозні ускладнення у разі застосування пошкоджених носіїв. Наприклад, якщо у разі збою мережі завантажити образ установочного диску із помилкою, то програмне забезпечення може бути все ж встановлено, але працювати воно буде досить нестабільно та знайти відповідну проблему буде дуже складно.

Після тестування носіїв буде завантажено графічну частину установнику (*Anaconda*) операційної системи (рис. А.2). У цій частині треба буде визначити параметри часу та дати, мережеві налаштування, файлову систему для подальшої роботи *CentOS* та ін.

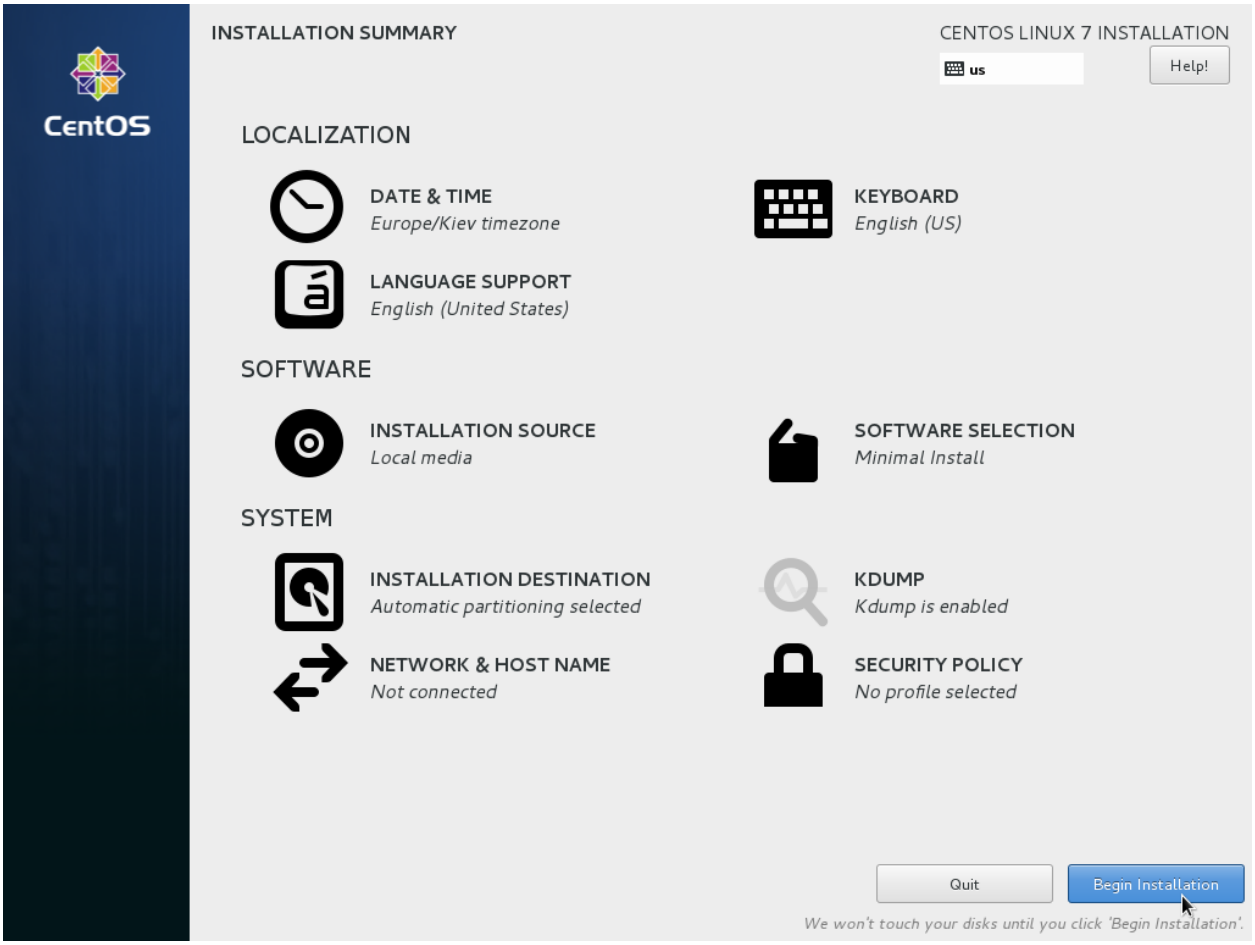

Рис. А.2. **Установка CentOS 7. Графічна частина установки**

Вибір розмітки диска треба виконувати дуже обережно, тому що помилкові дії можуть призвести до втрати попередніх даних на жорстких дисках. Зазвичай установка серверної операційної системи виконується на окремо виділеному комп'ютері, тому для її установки раціонально виділити весь обсяг накопичувача. Для цього етапу є зручним використання майстра розмітки диску. В умовах середовища віртуалізації налагодження дискового простору значно спрощується відповідно до його віртуального характеру.

Якщо не має причини виконувати специфічну розбивку носіїв, але не підходять рекомендовані налаштування майстра, то можна створити розділи: */swap*, */boot* та кореневий – /.

Розділ */boot* (мінімум 100 Мб), містить ядро операційної системи, а також файли початкового завантаження. Раціональним може бути створення окремого невеликого розділу для зберігання файлів завантаження (http://www.redhat.com/ docs/ manuals/ enterprise/ RHEL-5-manual/ ru-RU/ Installation_Guide/ index.html).

Розділ підкачування (*swap*) (не менше ніж 256 Мб) – використовується для організації віртуальної пам'яті. Дані попадають у розділ підкачування (*swap*), якщо системі для обробки даних не вистачає оперативної пам'яті. Рекомендовано обирати обсяг розділу підкачування, який дорівнює подвоєному обсягу оперативної пам'яті (RAM) у випадку, якщо обсяг RAM не перевищує 2 Гб, та який дорівнює тому ж обсягу для пам'яті 2 Гб і більше, але не менше 32 Мб. Таким чином, якщо: M = обсяг RAM, а S – обсяг розділу підкачування, то:

$$\begin{aligned} &\textit{if}\ \ M < 2 \\ &\qquad S = M * 2 \\ &\textit{else} \\ &\qquad S = M + 2 \end{aligned}$$

Можна легко визначити, що область підкачування для системи з фізичною RAM обсягом 2 Гб буде 4 Гб, для RAM обсягом 3 – 5 Гб. Великий обсяг розділу підкачування може мати сенс у разі планування збільшення обсягу оперативної пам'яті. Для систем із суттєво більшим обсягом оперативної пам'яті (більш 32 Гб) цілком можливий варіант створення меншого розділу підкачування (рівного обсягу RAM або менше).

Розділ *root* рекомендується створювати розміром 3 – 5 Гб. У цьому розділі буде розміщений кореневий каталог "/". У такому варіанті установки всі файли (крім розташованих у розділі */boot*) перебувають у кореневому розділі. Розділ розміром 3 Гб дозволить провести мінімальну установку, а кореневий розділ розміром 5 Гб буде задовольняти мінімальній конфігурації повної установки, що вміщує всі групи пакетів.

Зазвичай для операційної системи *Linux* використовуються різні типи файлових систем. Наприклад, розглянутий розділ *swap* застосовують для розділу підкачування. Для інших розділів можна застосовувати файлову систему, яка підтримує стандартні типи файлів (звичайні файли, каталоги, символічні посилання та інше).

Кращим вибором для збереження даних є файлова система *ext*3 чи *ext*4, що засновані на системі *ext*2 й мають перевагу над нею – ведення журналу. Це зменшує час відновлення файлової системи після збою, тому що необхідність у перевірці файлової системи командою *fsck* не потрібна.

Для застосування у якості файлової системи для *CentOS* можна обрати LVM (*Logical Volume Manager*), менеджер логічних томів – це система керування дисковим простором, що абстрагується від фізичних пристроїв. Вона дозволяє ефективно використовувати й легко управляти дисковим простором. LVM має гарну масштабованість, зменшує загальну складність системи (http://gazette.linux.ru.net/rus/articles/taleLinuxLVM.html). У логічних томів, створених за допомогою LVM, можна легко змінити розмір, а їх назви можуть нести більш поглиблене значення, на відміну від традиційних */dev/sda*, */dev/hda*.

Слід зазначити, що для серверної операційної системи може бути доцільним створення програмних RAID (англ. *Redundant Array of Independent /Inexpensive Disks*) масивів незалежних дисків. Такі масиви служать для підвищення надійності зберігання даних (наприклад, RAID 1) чи для підвищення швидкості читання/запису інформації (наприклад, RAID 0). Також, установку *CentOS* можна виконати на пристрій iSCSI. Протокол iSCSI (*Internet Small Computer System Interface*) базується на TCP/IP і розроблений для встановлення взаємодії та керування системами зберігання даних, серверами й клієнтами (http://www.ixbt.com/ storage/iscsi.shtml).

Після виконання розмітки файлових систем дискових накопичувачів програма установки *CentOS* буде запрошувати дані про настройки підключення мережевих карт. Якщо ці дані можна отримати автоматично від DHCP-серверу, то це значно спростить установку. Однак, для серверної системи скоріш за все треба буде вказати статичні налаштування для IP-адрес мережевих карт та повне ім'я майбутнього серверу. Після цього треба буде вказати часовий пояс, у якому знаходиться сервер та обов'язково вказати пароль адміністратора – *root*.

Наступним кроком установки буде визначення набору пакетів для серверної системи (рис. А.3). Спочатку треба обрати тип установки, наприклад: *Server*. Потім варто обирати групи пакетів.

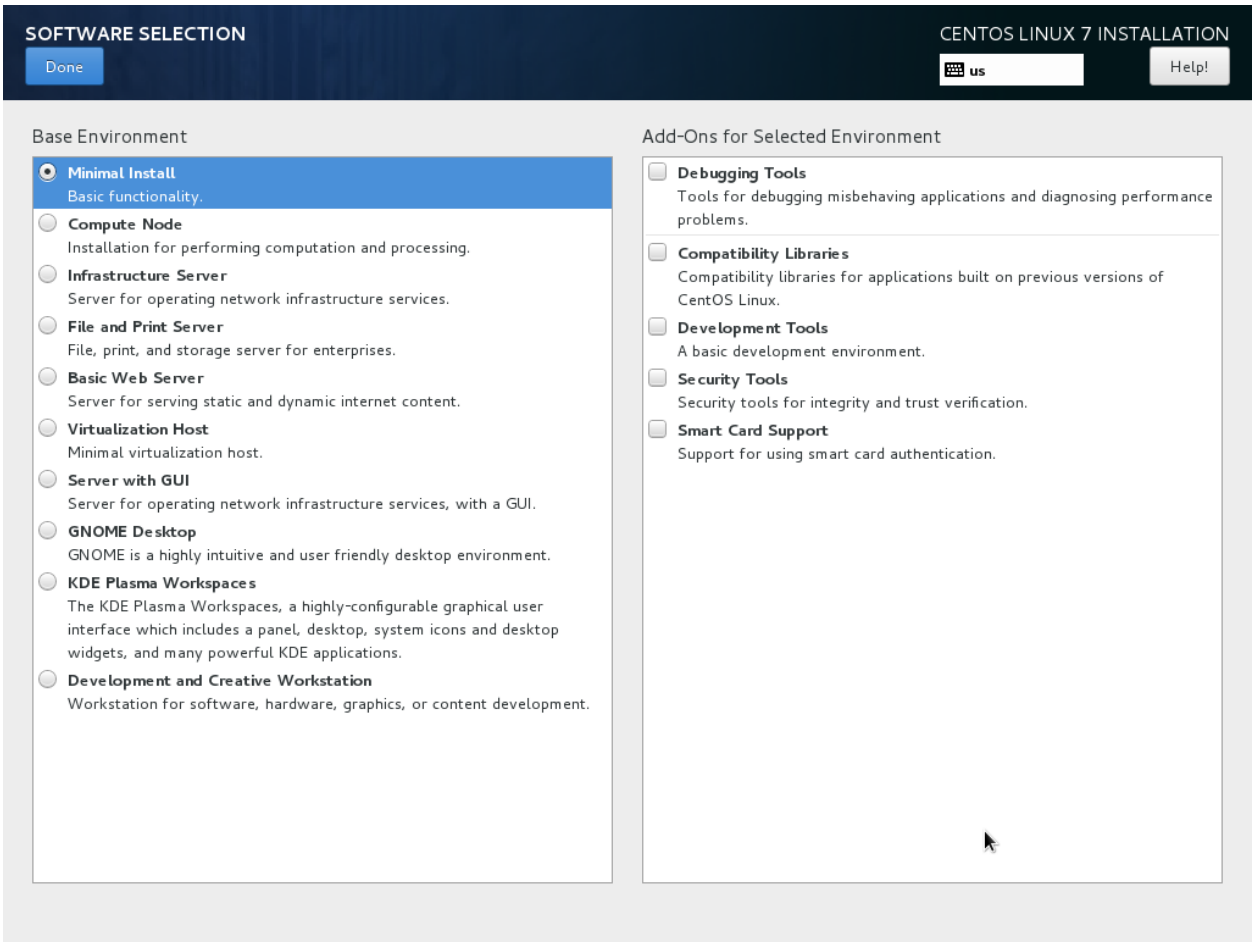

Рис. А.3. **Установка CentOS 7. Вибір пакетів**

Для кожної групи можна обрати певні пакети, наприклад, для *System Tools* можна обрати з *Optional packages* пакет *Midnight Commander* (*mc*), що є зручним файловим менеджером. Після вибору пакетів система установки перевірить залежності пакетів та перейде до наступного етапу. Тільки на цьому етапі буде фізично виконано форматування жорсткого диску, а потім всі компоненти операційної системи будуть установлені (рис. А.4).

Після установки програмного забезпечення, що було обрано до складу сервера, програма установки виконає перезавантаження системи, попередивши про те, що слід вийняти всі загрузочні носії. Після перезавантаження буде виконано перший запуск *CentOS*.

Слід відзначити, що важливим для першого запуску операційної системи є настройка мережевого екрану. Ці настройки можна виконати засобами майстра конфігурації системи – *Setup Agent*. Для цього в меню *Select Firewall configuration* треба установити необхідні параметри безпеки.

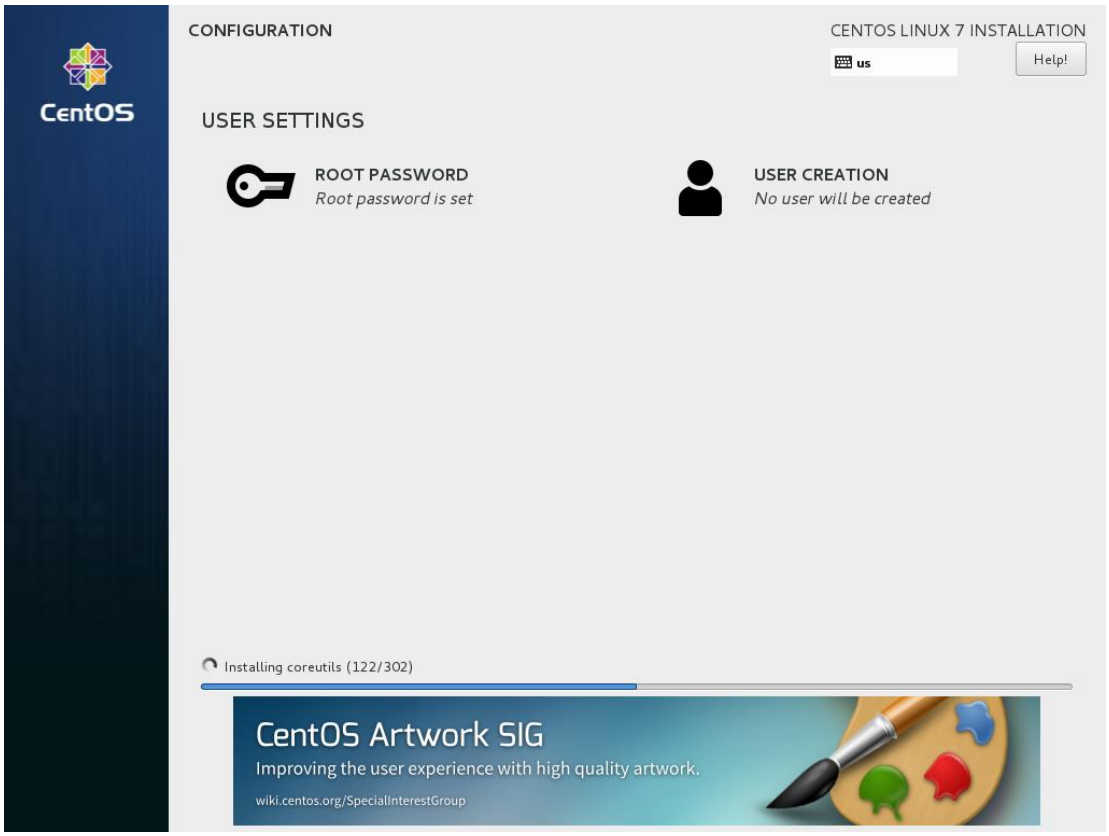

Рис. А.4. **Установка програмного забезпечення *CentOS* 7**

Важливим пунктом меню *Firewall configuration* є визначення параметрів SELinux (*Security-Enhanced Linux*) – *Linux* з поліпшеною безпекою. Це реалізація системи примусового контролю доступу, яка може працювати паралельно із класичною системою контролю доступу, а також входить у стандартне ядро *Linux*. Також для функціонування SELinux потрібні модифіковані версії деяких системних утиліт, які забезпечують підтримку нових функцій ядра, й підтримку з боку файлової системи.

У SELinux права доступу визначаються самою системою за допомогою встановлених спеціальних політик. Політики працюють на рівні системних викликів і застосовуються самим ядром (але можна реалізувати й на рівні додатка). SELinux діє після класичної моделі безпеки *Linux*. Іншими словами, через цю систему не можна дозволити те, що заборонене через права доступу користувачів/груп. Політики визначають за допомогою спеціальної гнучкої мови опису правил доступу. У більшості випадків правила SELinux "прозорі" для додатків та не потребують модифікації (за матеріалами з Вікіпедії).

Для виконання лабораторних робіт також є доцільним застосовувати установку *CentOS* у конфігурації сервера із графічним оточенням робочого простору (рис. А.5). Це дозволить застосовувати утиліти із графічною оболонкою, що буде зручним у рішенні завдань, наприклад, тестування приєднання клієнтської частини для розподіленого сховища даних.

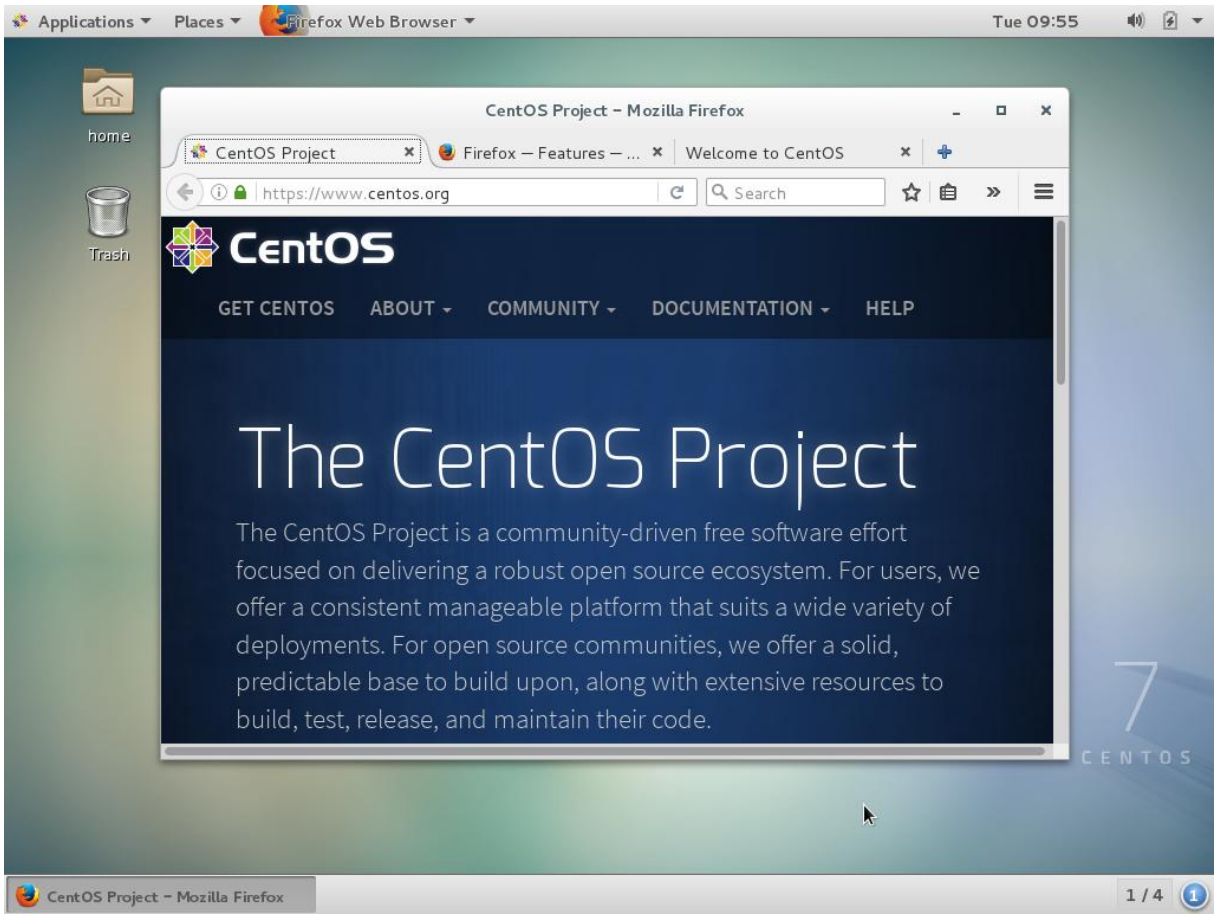

Рис. А.5. **Робота із *CentOS* 7 у графічному оточенні Gnome**

Наступним кроком конфігурування сервера є конфігурування мережевого інтерфейсу, наприклад: редагування файла */etc/sysconfig/network* щодо параметру: *hostname HOSTNAME=ceph-node1* дозволить встановити ім'я вузла.

Для встановлення параметрів мережевого контролеру, слід відредагувати файл */etc/sysconfig/network-scripts/ifcfg-eth0* та задати, наприклад режим DHCP:

ONBOOT=yes

BOOTPROTO=dhcp

Для конфігурування іншого мережевого інтерфейсу щодо роботи зі статичними параметрами, наприклад, для забезпечення комунікації по внутрішній мережі кластеру, слід відредагувати: */etc/sysconfig/network-scripts /ifcfg-eth1* та додати рядки:

ONBOOT=yes

BOOTPROTO=static

IPADDR=192.168.1.101

NETMASK=255.255.255.0

Для доступу до вузлів за доменними іменами, слід налаштувати файл */etc/hosts* та додати:

192.168.1.101 *ceph-node1*

192.168.1.102 *ceph-node2*

192.168.1.103 *ceph-node3*

Після установки мережевих налаштувань слід перезавантажити віртуальну машину, наприклад: *shutdown -r now* та після перевірки працездатності за протоколом SSH слід зайти на інші вузли та виконати відповідні налаштування: *ssh root*@192.168.1.102 та ін.

Для зручної роботи у середовищі кластера доцільно налаштувати доступ за протоколом SSH, що не буде вимагати введення пароля (http://www.tecmint.com/ssh-passwordless-login-using-ssh-keygen-in-5-easy-steps/). Для цього на вузлу слід згенерувати ключі шифрування:

ssh-keygen -t rsa

Потім на іншому вузлу додати каталог:

ssh username@192.168.0.11 mkdir -p .ssh та перенести до нього публічний ключ для авторизації:

cat .ssh/id_rsa.pub | *ssh username*@192.168.0.102 *'cat >> .ssh/authorized_keys'.*

Завершити налаштування слід за умови узгодження прав доступу:

ssh username@192.168.0.11 "*chmod* 700 *.ssh;*

chmod 640 *.ssh/authorized_keys*".

Після настройки мережевих засобів та параметрів безпеки можна виконати перезавантаження системи та перейти до виконання оновлення системи. Пакети в *CentOS* встановлюються за допомогою програми: *yum*. Для оновлення системи треба виконати команду: *yum update*.

Розгортання операційної системи *Ubuntu* у середовищі віртуалізації

Операційна система *Ubuntu Server* 16.04 має зручний інтерфейс установки (рис. Б.1). Для виконання лабораторних робіт бажано використовувати версію LTS (*Long Term Support*), тобто, яка має значний строк підтримки від розробників. Це надає можливості ознайомитися із особливостями системи, яка буде актуальною ще протягом декількох років.

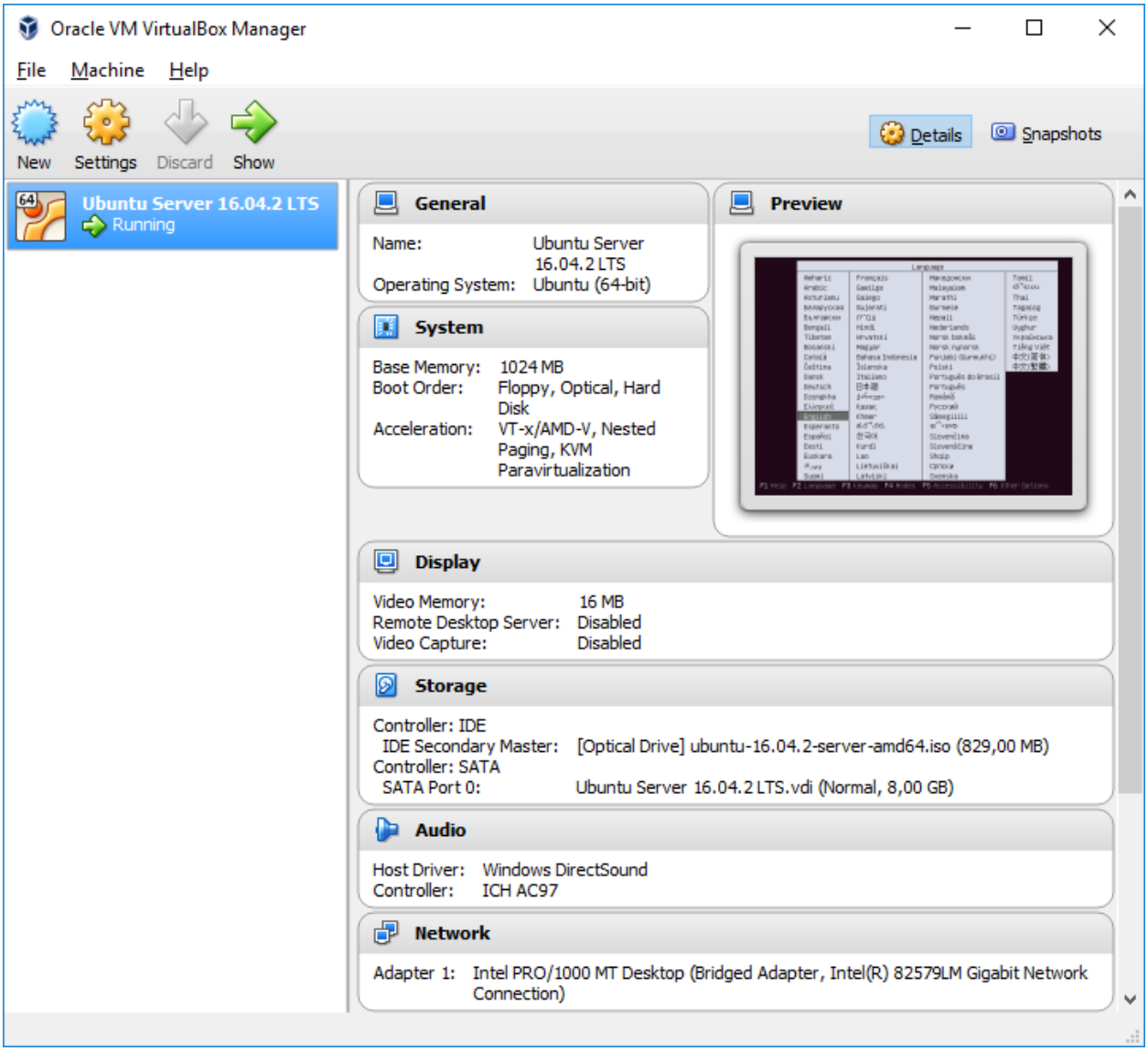

Рис. Б.1. **Встановлення *Ubuntu Server* у середовищі *VirtualBox***

Також слід відзначити, що доцільно вказати у налаштуваннях мережевого адаптера режим міст (*Bridged*).

У більшості випадків вихід у мережу буде здійснюватися за допомогою маршрутизатору, який скоріш за все буде мати сервіс DHCP. Це сукупно надає можливості у віртуальній машині отримати доступ до ресурсів Інтернет, поруч із можливістю спілкуватися із основною операційною системою чи іншими віртуальними машинами за протоколом *ssh*.

Установка визначеної операційної системи починається із вибору мови для подальшого розгортання, розкладки клавіатури тощо. Доцільно обрати, наприклад: *English* (US). Наступним кроком система установки за допомогою сервісу DHCP спробує визначити налаштування мережевого адаптера або виконає запит щодо необхідних налаштувань. Після цього слід ввести назву вузла (*hostname*), як наведено на рис. Б.2.

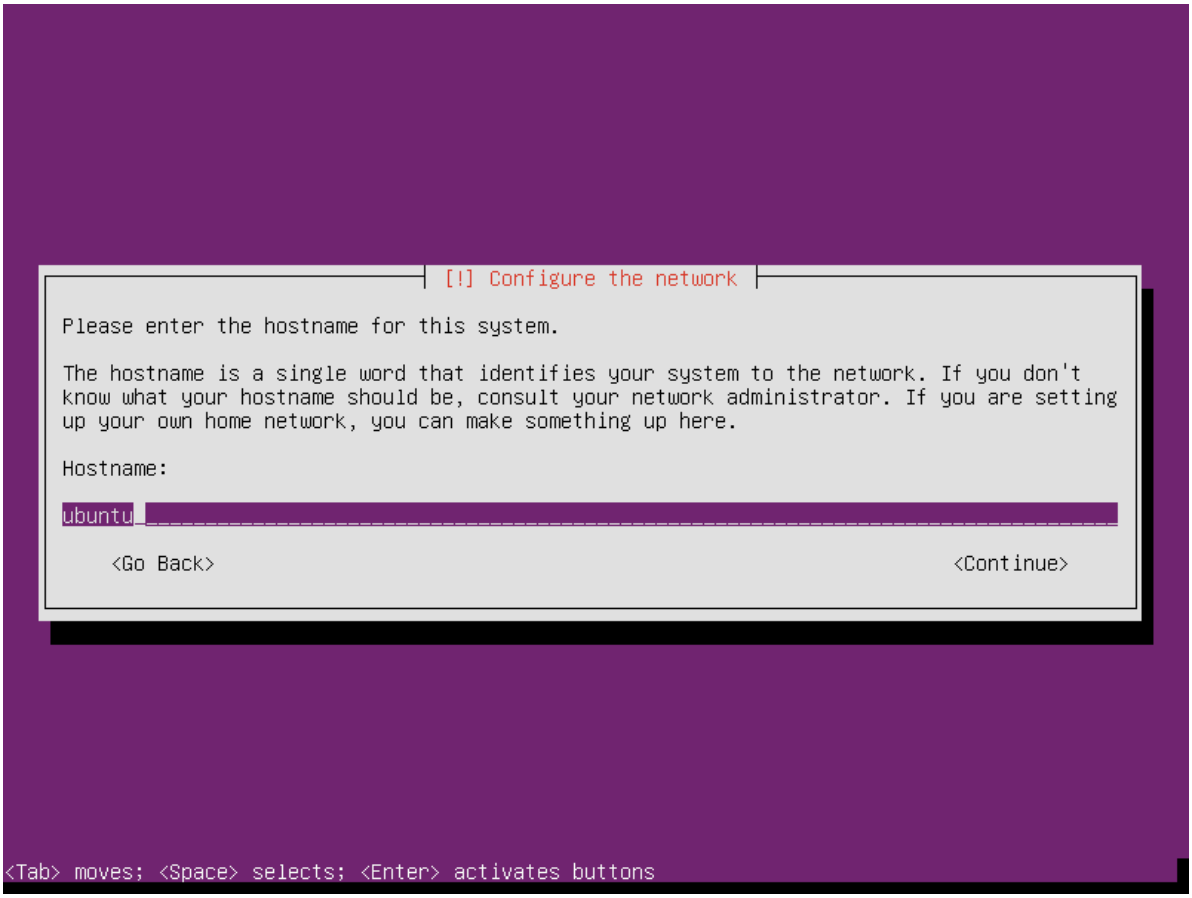

Рис. Б.2. **Визначення назви вузла**

Наступний крок – визначення імені користувача. Зазначимо, що за замовчуванням із міркування безпеки у дистрибутиві *Ubuntu* не має користувача *root*. Отримати відповідні повноваження за командою: *sudo su*.

Додавання суперкористувача у ході роботи із вже встановленою системою виконується за командою: *sudo passwd root*. Продовження установки, що виконується, буде супроводжуватися питанням щодо шифрування домашньої директорії. Для навчальних цілей не має сенсу у такому шифруванні. Наступний крок – визначення часової зони та розбивка накопичувача. Для формату віртуального дискового накопичувачу доцільно обрати автоматичний режим із LVM, що надає у подальшій роботі гнучкості в управлінні дисками та файловими системами (рис. Б.3).

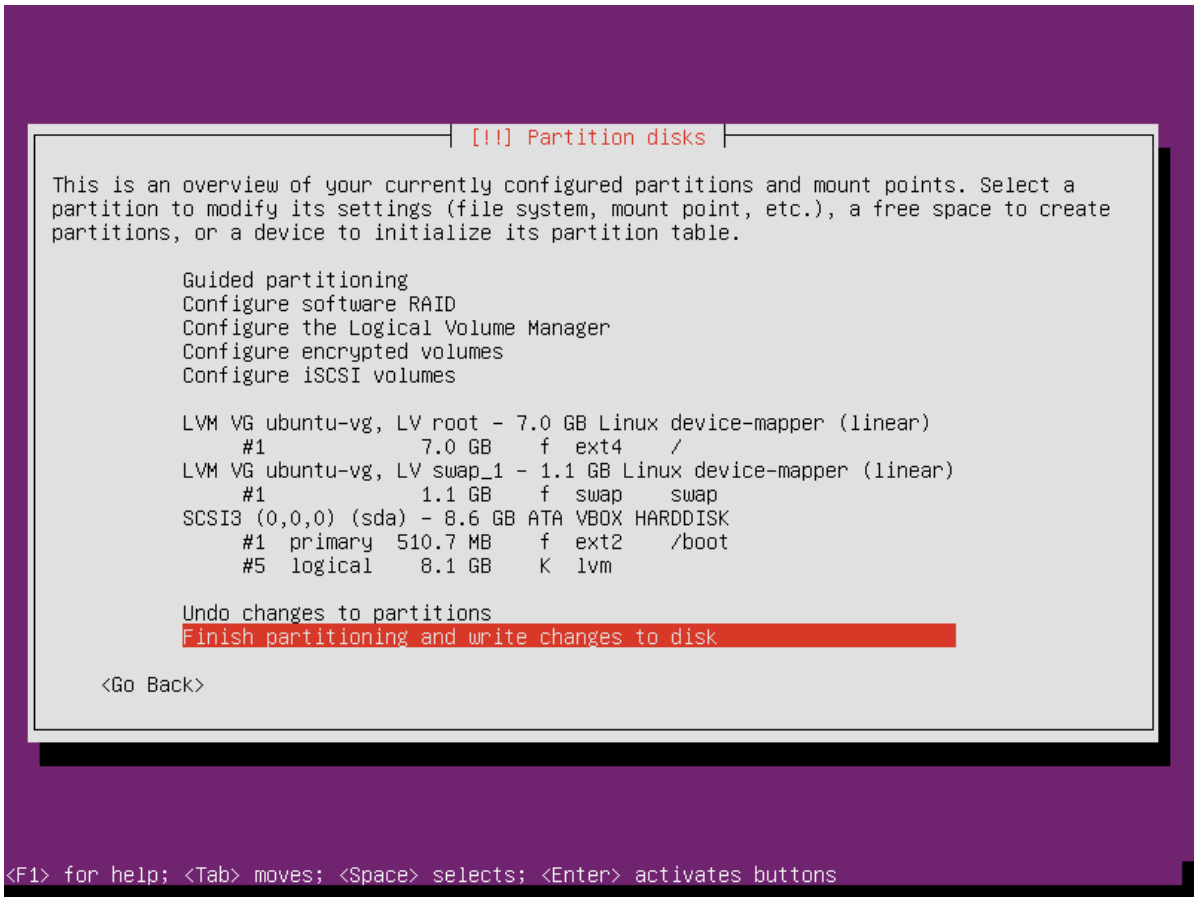

Рис. Б.3. **Розбивка віртуального накопичувача**

Наступним кроком система установки наводить результат автоматичного формування розділів та запрошує підтвердження виконання дій (у разі реальної системи слід відповідальніше відноситися до цього пункту, оскільки дії системи фактично видаляють дані з обраного накопичувача). Далі буде виконана розбивка та форматування накопичувача, а також будуть встановлені основні компоненти операційної системи.

У разі застосування проксі-серверу, слід вказати його адресу на поточному етапі розгортання системи. Також слід прийняти рішення щодо застосування механізму автоматичного оновлення системи. Оскільки розглядається серверна операційна система, то оновлення скоріш за все будуть тестувати у експериментальних умовах, а потім встановлюватися на сервер, тому слід обрати пункт: *No automatic updates*.

Для завершення установки слід обрати набір серверних компонентів (рис. Б.4).

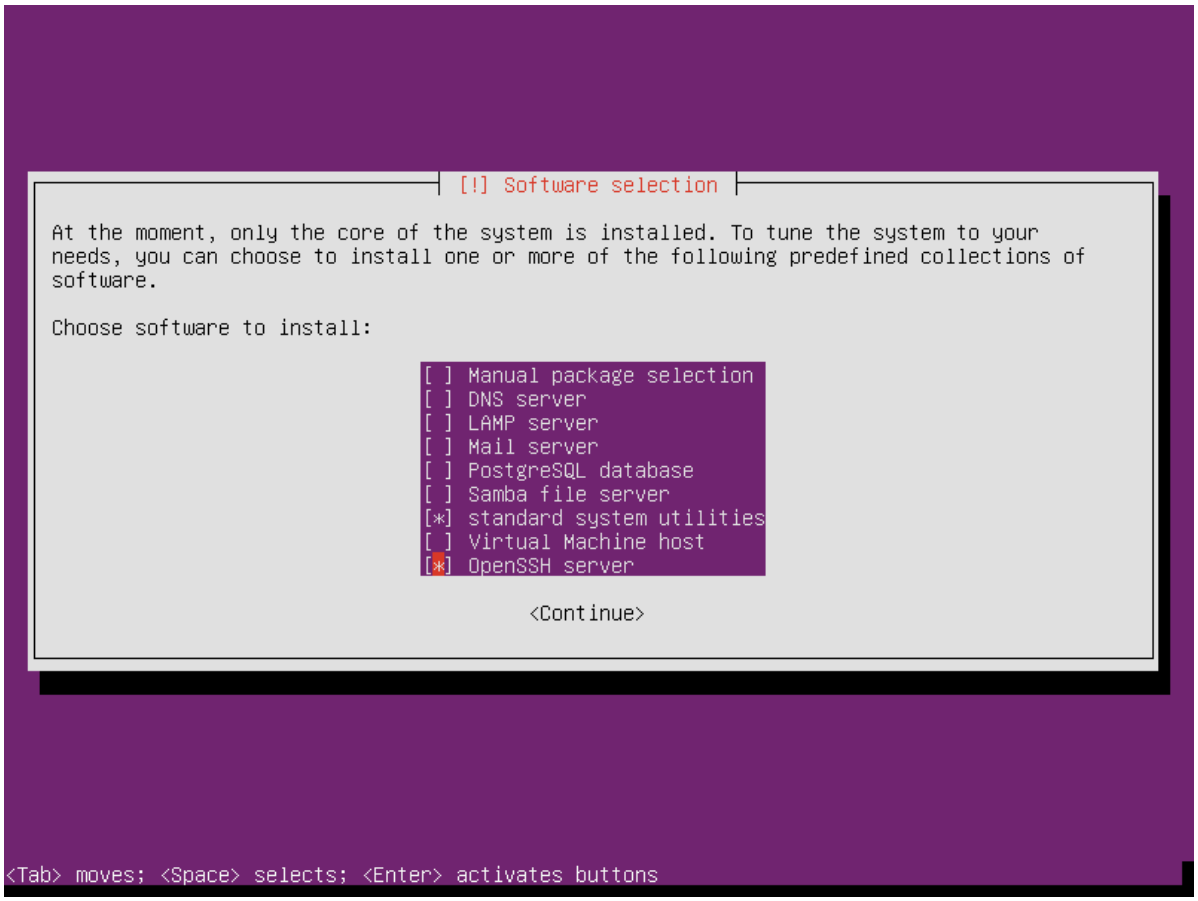

Рис. Б.4. **Встановлення сервісних компонентів**

Оскільки лабораторний практикум розраховано на вивчення особливостей серверних додатків, у якості додаткових компонентів доцільно встановити тільки сервіс *OpenSSH* для забезпечення зручного доступу до ресурсів віртуальної машини за протоколом *ssh*. А у якості клієнта протоколу *ssh*, наприклад у середовищі Windows, доцільно обрати *PuTTY* (http://www.chiark.greenend.org.uk/ ~sgtatham/putty/latest.html).

Завершується установка операційної системи *Ubuntu Server* встановленням завантажника GRUB. Після установки визначеної системи (рис. Б.5) слід видалити образ носія із установочними компонентами із конфігурації віртуальної машини.

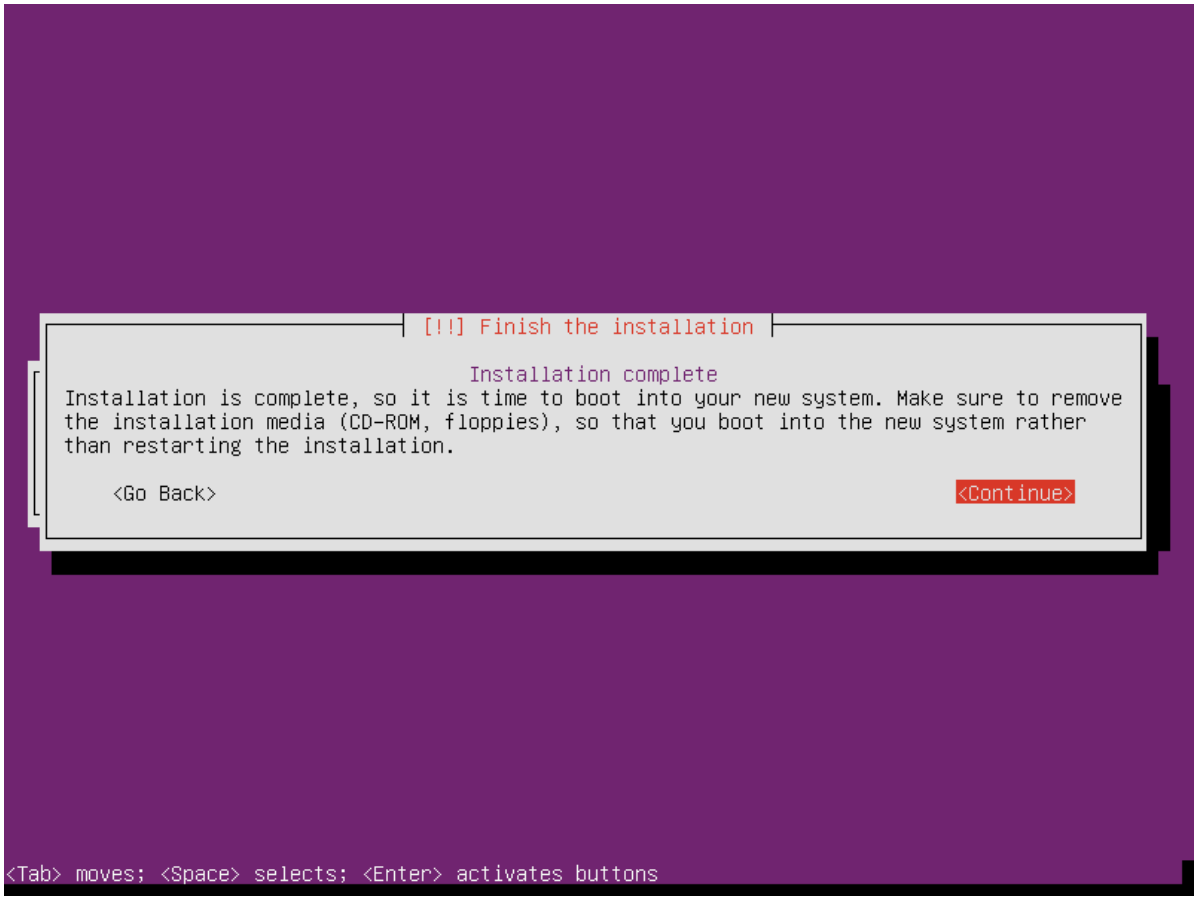

Рис. Б.5. **Останній крок установки *Ubuntu Server***

Для подальшого вивчення особливостей розподілених систем збереження даних доцільно виконати копіювання (клонування) створеної віртуальної машини та задіяти три-чотири вузли для виконання лабораторних робіт. Слід зазначити, що у середовищі віртуалізації слід обов'язково наявно вказати, що відповідні машини було скопійовано. Це надає можливості уникнути клонування систем, які будуть мати, наприклад, однакові MAC-адреси. Продовж роботи із клонованими системами може знадобитись виконати настроювання мережевих інтерфейсів. Наприклад, зміна імені вузла забезпечується: *sudo hostname new-host-name* та редагуванням: *sudo nano /etc/hostname, sudo nano /etc/hosts*.

Рекомендована література

Основна

1. Дейт К. Дж. Введение в системы баз данных / К. Дж. Дейт; пер. с англ. – Москва : ИД "Вильямс", 2005. – 1328 с.

2. Редмонд Э. Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL / Э. Редмонд, Дж. Р. Уилсон; пер. с англ. А. А. Слинкин – Москва : ДМК Пресс, 2013. – 384 с.

3. Тарасов С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. – Москва : СОЛОН-Пресс, 2015. – 320 с.

4. Уайт Т. Hadoop: Подробное руководство / Т. Уайт. – Санкт-Петербург : Питер, 2013. – 672 с.

5. MySQL. Оптимизация производительности / Б. Шварц, П. Зайцев, В. Ткаченко и др. – 2-е изд. – Санкт-Петербург : Символ-Плюс, 2010. – 832 с.

6. Unix и Linux. Руководство системного администратора / [Э. Немет, Г. Снайдер, Т. Хейн и др.] – Москва : ИД "Вильямс", 2012. – 1312 с.

Додаткова

7. Бьюли А. Изучаем SQL / А. Бьюли ; пер. с англ. – Санкт-Петербург : Символ-Плюс, 2007. – 312 с.

8. Бэнкер К. MongoDB в действии / К. Бэнкер ; пер. с англ. А. А. Слинкина – Москва : ДМК Пресс, 2012. – 394 с.

9. Грофф Дж. Р. SQL: полное руководство / Дж. Р. Грофф, П. Н. Вайнберг, Э. Дж. Оппель ; пер. с англ. – 3-е изд. – Москва : ИД "Вильямс", 2015. – 960 с.

10. Уорсли Дж. PostgreSQL. Для профессионалов / Дж. Уорсли, Дж. Дрейк. – Санкт-Петербург : Питер, 2003. – 496 с.

11. Фаулер М. NoSQL: новая методология разработки нереляционных баз данных / М. Фаулер, П. Дж. Садаладж ; пер. с англ. – Москва : ИД "Вильямс", 2013. – 192 с.

12. Monitoring with Ganglia / M. Massie, B. Li, B. Nicholes, V. Vuksan. – O'Reilly Media, Inc. – Sebastopol, CA. – 2012. – 256 p.

13. Wojciech K. Learning Nagios 4 / K. Wojciech. – Packt Publishing. – Birmingham. – 2014. – 400 p.

Інформаційні ресурси

14. Алексієв В. О. Інформаційно-комунікаційна технологія розроблення транспортно-інформаційного порталу / В. О. Алексієв, В. С. Наумов, М. А. Суховаров, Г. О. Васютіна // Інформаційні технології і мехатроніка. Освіта, наука та працевлаштування: зб. наук. пр. – Харків: Стиль-Издат, 2016. – С. 9–16. – Режим доступу : http://www.repository.hneu.edu.ua/jspui /handle/123456789/13393.

15. Бездисковая загрузка по технологии iSCSI на базе ОС Windows [Электронный ресурс] / Хабрахабр, 2014. – Режим доступа : https://habrahabr.ru/post/244661/.

16. Бенинкоса В. Ganglia и Nagios: Часть 1. Мониторинг коммерческих кластеров с помощью Ganglia [Электронный ресурс] / В. Бенинкоса // IBM developerWorks, 2009. – Режим доступа : http://www.ibm.com/ developerworks/ru/library/l-ganglia-nagios-1/.

17. Бенинкоса В. Ganglia и Nagios: Часть 2. Мониторинг коммерческих кластеров с помощью Nagios [Электронный ресурс] / В. Бенинкоса // IBM developerWorks, 2009. – Режим доступа : https://www.ibm.com/developerworks/ ru/library/l-ganglia-nagios-2/.

18. Васильев А. Ю. Работа с PostgreSQL: настройка и масштабирование [Электронный ресурс] / А. Ю. Васильев. – Режим доступа : http://postgresql.leopard.in.ua/.

19. Вопросы создания и обслуживания программного RAID-массива в операционной системе Linux [Электронный ресурс]. – Режим доступа : http://xgu.ru/wiki/mdadm.

20. Гудвилл Дж. Memcached и Grails. Часть 1: Установка и применение memcached [Электронный ресурс] / Дж. Гудвилл. – IBM developerWorks. – Режим доступа : http://www.ibm.com/developerworks/ru/library/ j-memcached1/.

21. Деза А. Большое, быстрое, стабильное и недорогое хранилище данных [Электронный ресурс] / А. Деза // IBM developerWorks. – 2012. – Режим доступа : http://www.ibm.com/ developerworks/ ru/library/os-zfsraidz/.

22. Джонс М. Т. Linux и системы хранения данных [Электронный ресурс] / М. Т. Джонс // IBM developerWorks, 2011. – Режим доступа : http://www.ibm.com/ developerworks/ru/library/l-linux-storage/.

23. Документация PostgreSQL и Postgres Pro [Электронный ресурс] / Компания "Постгрес Профессиональный", 2016. – Режим доступа : https://postgrespro.ru/docs/.

24. Зорин А. Быстрое развертывание решения на основе пакета Nagios 3 для мониторинга распределенных систем и сетей [Электронный ресурс] / А. Зорин // IBM developerWorks. – 2011. – Режим доступа : http://www.ibm.com/developerworks/ru/library/l-nagios/.

25. Знакомство с хранилищем Ceph в картинках [Электронный ресурс] / Хабрахабр, 2016. – Режим доступа : https://habrahabr.ru/ post/313644/.

26. "Идеальный" кластер. Часть 3.1 Внедрение MySQL Multi-Master кластера [Электронный ресурс] / Хабрахабр, 2015. – Режим доступа : https://habrahabr.ru/post/253869/.

27. Климонтович В. Apache Hadoop (ADD-2010) [Электронный ресурс] / В. Климонтович. – Режим доступа : http://lib.custis.ru/ Apache_ Hadoop_ (Владимир_Климонтович_на_ADD-2010).

28. Киви К. Г. Управление логическими томами [Электронный ресурс] / К. Г. Киви // IBM developerWorks, 2008. – Режим доступа : http://www.ibm.com/ developerworks/ru/library/l-lvm2/.

29. Как устроена apache cassandra [Электронный ресурс] / Хабрахабр. – Режим доступа : http://habrahabr.ru/post/155115/.

30. Лейрд К. Увеличение возможностей Nagios с помощью плагинов собственной разработки [Электронный ресурс] / К. Лейрд, К. Войцех IBM developerWorks, 2009. – Режим доступа : https://www.ibm.com/developerworks/ ru/library/au-nagios/.

31. Лекции Техносферы. 2 семестр. Методы распределенной обработки больших объемов данных в Hadoop [Электронный ресурс] / Блог компании Mail.Ru Group. – Режим доступа : http://habrahabr.ru/company/mailru/ blog/ 258045/.

32. Лекции Технопарка. 3 семестр. Проектирование высоконагруженных систем [Электронный ресурс] / Блог компании Mail.Ru Group. – Режим доступа : http://habrahabr.ru/ company/ mailru/ blog/ 254843/.

33. Минухин С. В. Информационная технология для планирования заданий на вычислительных кластерах распределенной системы на основе интеграции сервисов удаленного доступа / С. В. Минухин // Системи обробки інформації. – 2015. – Вип. 12. – С. 134–139. – Режим доступу : http://nbuv.gov.ua/UJRN/soi_2015_12_33.

34. Настройка nagios для проверки в труднодоступных местах (NRPE) (nagios monitoring nrpe) [Электронный ресурс] // OpenNET. – 2009. – Режим доступа : https://www.opennet.ru/base/net/nagios_nrpe.txt.html.

35. Неубиваемый Postgresql cluster внутри Kubernetes cluster [Электронный ресурс] / Хабрахабр, 2016. – Режим доступа : https://habrahabr.ru/post/301370/.

36. Новый aggregation framework в MongoDB 2.1 [Электронный ресурс] / Хабрахабр, 2012. – Режим доступа : https://habrahabr.ru/post/139643/.

37. Обработка данных NBA за 30 лет с помощью MongoDB Aggregation [Электронный ресурс] / Хабрахабр, 2014. – Режим доступа : https://habrahabr.ru/ company/itinvest/blog/246317/.

38. Отказоустойчивый кластер Master-Slave на PostgreSQL [Электронный ресурс] / Хабрахабр, 2013. – Режим доступа : https://habrahabr.ru/post/188096/.

39. Очень большой Postgres [Электронный ресурс] / Хабрахабр, 2015. – Режим доступа : https://habrahabr.ru/post/253017/.

40. Перера С. Погружение в СУБД Apache Cassandra [Электронный ресурс] / С. Перера // IBM developerWorks. – Режим доступа : https://www.ibm.com/ developerworks/ ru/ library/ os-apache-cassandra/.

41. Программный RAID в Linux. Тестирование и мониторинг [Электронный ресурс] / OpenSource в заметках, 2012. – Режим доступа : http://ashep.org/2012/programmnyj-raid-v-linux-testirovanie-i-monitoring/.

42. Распределенные базы и хранилища данных : Электронный учебник / Н. Аносова, О. Бородин, Е. Гаврилов и др. – НОУ "ИНТУИТ" [Электронный ресурс]. – Режим доступа : http://www.intuit.ru/ studies/courses/1145/214/info.

43. Распределенные файловые системы. Технологии хранения и обработки больших объемов данных / Computer Science Center [Электронный ресурс]. – Режим доступа : https://compscicenter.ru/courses/big-data/ 2015-spring/ classes/1117/.

44. Распределённые вычисления поверх Ceph RADOS и AsyncMessenger [Электронный ресурс] / Хабрахабр, 2017. – Режим доступа : https://habrahabr.ru/post/330908/.

45. Развертывание кластера Postgres-xl для чайников [Электронный ресурс] / Хабрахабр, 2015. – Режим доступа : https://habrahabr.ru/post/261457/.

46. Распределенная файловая система Ceph FS за 15 минут [Электронный ресурс] / Хабрахабр, 2013. – Режим доступа : https://habrahabr.ru/post/179823/.

47. Руководство пользователя Zabbix [Электронный ресурс]. – Режим доступа : http://www.zabbix.com/ru/documentation.php.

48. Снастин А. Создание программного RAID-массива на Linux-платформе [Электронный ресурс] / А. Снастин // IBM developerWorks, 2011. – Режим доступа : http://www.ibm.com/developerworks/ru/library/l-soft-raid/.

49. Системы хранения данных [Электронный ресурс] / Вики-учебник, 2016. – Режим доступа : https://ru.wikibooks.org/wiki/Системы_хранения_данных.

50. Создание надёжного iSCSI-хранилища на Linux, часть 1 [Электронный ресурс] / Хабрахабр, 2014. – Режим доступа : https://habrahabr.ru/post/209460/.

51. Создание надёжного iSCSI-хранилища на Linux, часть 2 [Электронный ресурс] / Хабрахабр, 2014. – Режим доступа : https://habrahabr.ru/post/209666/.

52. Стратегии масштабирования MySQL [Электронный ресурс] // Блог о разработке высокопроизводительных PHP приложений.– 2012.– Режим доступа: http:// www.phphighload.com/ 2012/ 10/ mysql-scaling-strategies.html.

53. Строим свое собственное отказоустойчивое облако на базе OpenNebula с Ceph, MariaDB Galera Cluster и OpenvSwitch [Электронный ресурс] / Хабрахабр, 2015. – Режим доступа : https://habrahabr.ru/post/270187/.

54. Тестируем распределение контента в GlusterFS / Хабрахабр [Электронный ресурс]. – Режим доступа : http://habrahabr.ru/post/251931/.

55. Установка и настройка распределённой файловой системы GlusterFS в ОС CentOS 7.0 / Блог OS CONFIG [Электронный ресурс]. – Режим доступа : http://osc.dondub.com/articles/2015/04/article_852.

56. Установка и настройка iSCSI Target & Initiator на Centos/RHEL 5/6 [Электронный ресурс] / LinuxSpace, 2013. – Режим доступа : https://www.linuxspace.org/ archives/5085.

57. Установка NRPE на CentOS 7 [Электронный ресурс] / IT Stuffs, 2016. – Режим доступа : http://itstuff.info/linux_unix/ install_nrpe_on_centos_7/.

58. Черепенин С. PXE [Электронный ресурс] / С. Черепенин, И. Чубин // Xgu.ru, 2013. – Режим доступа : http://xgu.ru/wiki/PXE.

59. Яковлев С. MySQL и PostgreSQL. Часть 5. Масштабирование PostgreSQL [Электронный ресурс] / С. Яковлев // IBM developerWorks, 2010. – Режим доступа : https://www.ibm.com/developerworks/ru/library/os-mysql-postgresql/05/.

60. Ceph: Cloud Storage без компромиссов [Электронный ресурс] / Хабрахабр, 2014. – Режим доступа : https://habrahabr.ru/company/performix/blog/218065/.

61. Ceph: Распределенная файловая система петабайтных масштабов для Linux [Электронный ресурс] / Виртуальная энциклопедия "Linux по-русски", 2010. – Режим доступа : http://rus-linux.net/nlib.php?name=/MyLDP/file-sys/ceph/ceph.html.

62. Map-Reduce на примере MongoDB [Электронный ресурс] / Хабрахабр, 2013. – Режим доступа : https://habrahabr.ru/post/184130/.

63. Memcached и PHP ликбез [Электронный ресурс] / Хабрахабр. – Режим доступа : http://habrahabr.ru/post/108274/.

64. MongoDB от теории к практике. Руководство по установке кластера mongoDB [Электронный ресурс] / Хабрахабр, 2014. – Режим доступа : https://habrahabr.ru/post/217393/.

65. MongoDB Sharded Cluster на Centos 6.5 [Электронный ресурс] / Хабрахабр, 2014. – Режим доступа : https://habrahabr.ru/post/227395/.

66. Percona XtraDB Cluster. Установка и тестирование [Электронный ресурс] / Хабрахабр, 2012. – Режим доступа : https://habrahabr.ru/post/152969/.

67. A Complete Guide to FreeNAS Hardware Design, Part I: Purpose and Best Practices – FreeNAS [Electronic resource] / FreeNAS, 2015. – Access mode : http://www.freenas.org/blog/a-complete-guide-to-freenas-hardware-design-part-i-purpose-and-best-practices/.

68. A Complete Guide to FreeNAS Hardware Design, Part II: Hardware Specifics – FreeNAS [Electronic resource] / FreeNAS, 2015. – Access mode : http:// www.freenas.org/ blog/ a-complete-guide-to-freenas-hardware-design-part-ii-hardware-specifics/.

69. A Complete Guide to FreeNAS Hardware Design, Part III: Pools, Performance, and Cache – FreeNAS [Electronic resource] / FreeNAS, 2015. – Access mode : http://www.freenas.org/blog/a-complete-guide-to-freenas-hardware-design-part-iii-pools-performance-and-cache/.

70. A Complete Guide to FreeNAS Hardware Design, Part IV: Network Notes & Conclusion – FreeNAS [Electronic resource] / FreeNAS, 2015. – Access mode : http://www.freenas.org/blog/a-complete-guide-to-freenas-hardware-design-part-iv-network-notes-conclusion/

71. Connecting Windows 7 to an iSCSI SAN [Electronic resource] / TechGenix Ltd., 2009. – Access mode : http:// www.windowsnetworking.com/articles-tutorials/ windows-7/ Connecting-Windows-7-iSCSI-SAN.html.

72. Configure Software RAID on Linux [Electronic resource] / Rick Claus, Microsoft Azure, 2016. – Access mode : https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-configure-raid/

73. Depardon B. Analysis of Six Distributed File Systems / B. Depardon, G. Le Mahec, C. Séguin. – Research Report. – 2013. – 44 p. [Electronic resource]. – Access mode : https://hal.archives-ouvertes.fr/hal-00789086.

74. FreeNAS User Guide [Electronic resource]. – Access mode : http://doc.freenas.org/.

75. FreeNAS 10: A developer's perspective [Electronic resource] / FreeNAS, 2015. – Access mode : http://www.freenas.org/blog/freenas-10-a-developers-perspective/

76. Singh K. Ceph Cookbook [Electronic resource] / Packt Publishing, 2016. – Access mode : https://www.packtpub.com/mapt/book/ virtualization_ and_cloud/9781784393502.

77. Singh K. Learning Ceph [Electronic resource] / Packt Publishing, 2015. – Access mode : https://www.packtpub.com/mapt/book/ virtualization_ and_cloud/9781783985623.

78. The MongoDB 3.2 Manual [Electronic resource] / MongoDB, 2016. – Access mode : https://docs.mongodb.com/manual/.

Методичне забезпечення

79. Алексієв В. О. Застосування GRID-технології у транспортному ВНЗ : навч.-метод. посіб. / В. О. Алексієв.– Харків : ХНАДУ, 2008. – 208 с.

80. Алексієв В. О. Інформаційний розвиток порталу віртуального управління процесами транспортного обслуговування / В. О. Алексієв, О. П. Алексієв // Інформаційні технології: проблеми та перспективи : монографія / за заг. ред. В. С. Пономаренка. – Харків : Вид-во: Рожко С. Г., 2017.– С. 32–47.

81. Методы и модели планирования ресурсов в GRID-системах : монография / В. С. Пономаренко, С. В. Листровой, С. В. Минухин и др. ; Харк. нац. экон. ун-т. – Харьков : ИД "ИНЖЭК", 2008. – 407 с.

82. Методи та моделі розроблення комп'ютерних систем і мереж : монографія / В. С. Пономаренко, С. В. Мінухін, С. В. Кавун та ін. – Харків : Вид. ХНЕУ, 2008. – 315 с.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

РОЗПОДІЛЕНІ СХОВИЩА ДАНИХ

Методичні рекомендації до виконання лабораторних робіт

для студентів спеціальності 122 "Комп'ютерні науки та інформаційні технології" другого (магістерського) рівня

Самостійне електронне текстове мережеве видання

Укладач **Алексієв** Володимир Олегович

Відповідальний за видання *О. Г. Руденко*

Редактор *К. Л. Бикова*

Коректор *К. Л. Бикова*

План 2017 р. Поз. 100 ЕВ. Обсяг 66 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру ***ДК № 4853 від 20.02.2015 р.***