МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ ТА ХМАРНІ ТЕХНОЛОГІЇ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт та самостійної роботи
для студентів спеціальності 123 – «Комп'ютерна інженерія»

Обговорено і рекомендовано
на засіданні кафедри
інформаційних та комп'ютерних систем
Протокол № 2
від 27 вересня 2017 р.

Чернігів ЧНТУ 2018

Розподілені обчислення та хмарні технології. Методичні вказівки до виконання лабораторних робіт та самостійної роботи для студентів спеціальності 123 – «Комп'ютерна інженерія». / Укл.: Пріла О.А., Андрущенко Р.Б. – Чернігів: ЧНТУ, 2018. - 23 с., укр. мовою.

Укладачі: ПРІЛА ОЛЬГА АНАТОЛІЇВНА, доцент кафедри інформаційних та комп'ютерних систем
АНДРУЩЕНКО РОМАН БОГДАНОВИЧ, асистент кафедри інформаційних та комп'ютерних систем

Відповідальний за випуск: ЗАЙЦЕВ СЕРГІЙ ВАСИЛЬОВИЧ, завідувач кафедри інформаційних та комп'ютерних систем, доктор технічних наук, доцент

Рецензент: НІКІТЕНКО ЄВГЕНІЙ ВАСИЛЬОВИЧ, кандидат фізико-математичних наук, доцент кафедри інформаційних та комп'ютерних систем Чернігівського державного технологічного університету

Зміст

Вступ

Курс «Розподілені обчислення та хмарні технології» присвячений вивченню сучасних підходів до проектування розподілених систем та інструментальних засобів їх реалізації. Дисципліна охоплює сучасні підходи, стандарти та технології для розподілених обчислень.

Розглядаються розподілені системи управління базами даних та технології проектування розподілених обчислювальних систем, включаючи механізми реплікації, виконання розподілених транзакцій та планування завдань.

Вивчаються патерни проектування розподілених прикладних застосувань, а саме мікросервіси, кластеризація та автоматизоване розгортання компонентів застосування.

У методичних вказівках представлені різні аспекти проектування та розробки розподілених обчислень та систем. Розглядаються типові підходи до проектування і реалізації розподілених систем обробки та зберігання даних, а також практичні особливості використання відповідних технологій. Курс орієнтований на використання вільного програмного забезпечення.

Під час виконання лабораторних робіт вивчаються практичні особливості використання таких програмних засобів: Arc Nordugrid, Apache Hadoop, PBS Torque, Mongo DB, RethinkDB, OpenNebula. Студент може самостійно обирати альтернативні засоби реалізації відповідно до завдання лабораторної роботи.

Звіт з лабораторних робіт має бути чітко та якісно оформлений та включати такі розділи: постановка задачі; процес встановлення та налаштування програмних засобів, що використовуються; результати проектування розподілених обчислень задачі, що вирішується; результати експериментів з оцінки ефективності виконання обчислювальної задачі; висновки.

Методичні вказівки можуть використовуватися під час виконання кваліфікаційної роботи магістра.

1 Лабораторна робота №1 Виконання розподілених обчислень у грід-середовищі на базі ARC Nordugrid

1.1 Мета роботи

Ознайомитись з архітектурою та принципами функціонування грід-середовища на базі ARC Nordugrid. Навчитись конфігурувати, виконувати задачі в ARC Nordugrid та отримувати результати виконання.

1.2 Теоретичні відомості

Грід-обчислення є формою розподілених обчислень, де віртуальний суперкомп'ютер представлений у вигляді слабозв’язаних гетерогенних обчислювальних ресурсів, об'єднаних за допомогою глобальної мережі. Грід-інфраструктура використовується для виконання високопродуктивних обчислень в різних наукових та прикладних галузях (фізика, медичні дослідження, економічне прогнозування, єкологічний моніторинг та ін.).

Існує декілька проектів щодо розробки проміжного програмного забезпечення грід (ППЗ), яке розробляється в межах стандарту EMI (European Middleware Initiative, https://www.dcache.org/manuals/EMI_FACT-SHEET-1_4.pdf), а саме:

- Arc Nordugrid (http://www.nordugrid.org/);
- globus toolkit (http://toolkit.globus.org/grid_software/);
- gLite (http://grid-deployment.web.cern.ch/grid-deployment/glite-web/);
- UNICORE (https://en.wikipedia.org/wiki/UNICORE).

Українська національна грід-інфраструктура (http://ung.in.ua/) використовується для вирішення задач великої розмірності різних наукових галузей та побудована на базі ППЗ Arc Nordugrid.

1.3 Порядок виконання роботи

Під час виконання лабораторної роботи необхідно:

- ознайомитись з архітектурою грід-середовища на базі проміжного програмного забезпечення ARC Nordugrid, ресурсами українського національного грід-середовища та існуючими віртуальними організаціями: http://www.nordugrid.org/monitor/loadmon.php;
- розробити розподілений алгоритм виконання обчислювальної задачі;
- встановити arc nordugrid client (http://www.nordugrid.org/documents/arc-client-install.html) та ознайомитися з документацією (http://www.nordugrid.org/documents/arc-ui.pdf);
- завантажити проксі-сертифікат (тимчасовий доступ буде надано під час виконання лабораторної роботи);
- розробити специфікацію задачі для виконання у грід-середовищі у xrsl-форматі: http://www.nordugrid.org/documents/xrsl.pdf;

- запустити обчислювальну задачу на виконання у грід-середовищі;
- список доступних ресурсів: http://www.nordugrid.org/monitor/loadmon.php,VO=Ukraine,VO=medgrid:
  grid.isma.kharkov.ua
  arc.univ.kiev.ua
  arc.matmoden.kiev.ua
  felix.iap.sumy.org
  west.icmp.lviv.ua
  arc.imbg.org.ua
  cluster.immsp.kiev.ua
  inparcom.kiev.ua
  grid.inparcom.kiev.ua
- переглянути стан виконання задачі за допомогою команди arcstat;
- завантажити результат виконання задачі за допомогою arcget;
- оцінити ефективність виконання розподіленого алгоритму у грід-середовищі.

УВАГА! Всі наступні дії виконуються виключно під акаунтом поточного користувача в системі (не **root**).

Встановлення клієнта. Приклад для ОС Ubuntu.

1) Реєстрація ключа репозиторію в системі.

```
$ sudo wget -q -O
https://dist.eugridpma.info/distribution/igtf/current/GPG-KEY-
EUGridPMA-RPM-3 | apt-key add -
```

2) Реєстрація репозиторію в системі: створити файл nordugrid.list в каталозі: `/etc/apt/sources.d/`

```
$ sudo touch /etc/apt/sources.d/nordugrid.list
```

Додати репозиторії в цей файл:

```
$ sudo nano /etc/apt/sources.d/ ;запуск редактора nano

deb http://repository.egi.eu/sw/production/cas/1/current egi-igtf core
deb [trusted=yes] http://download.nordugrid.org/repos/15.03/ubuntu/
                                                    artful main
deb [trusted=yes] http://download.nordugrid.org/repos/15.03/ubuntu/
                                                    artful-updates main
```

Для збереження змін натиснути CTRL+X, потім Y та Enter.

УВАГА! Для різних версій ОС використовуються різні репозиторії. Актуальний перелік знаходиться на офіційному сайті ARC Nordugrid.

3) Встановлення клієнта:

```
$ sudo apt-get update
$ sudo apt-get install ca-policy-egi-core
$ sudo apt-get install nordugrid-arc-client-tools
```

```
$ sudo apt-get install nordugrid-arc-client nordugrid-arc-plugins-
needed nordugrid-arc-plugins-globus ca-policy-igtf-classic ca-
policy-igtf-mics ca-policy-igtf-slcs fetch-crl
```

4) В домашньому каталозі створити підкаталоги (якщо їх нема):

```
/home/<username>/.arc
/home/<username>/.globus
```

5) У разі, якщо ці каталоги відсутні, створити файл client.conf (в раніше створених каталогах .arc та .globus). Модифікувати файли наступним чином:

```
[common]
proxypath=/home/<username>/.globus/cert.bin
```

6) Отримати у викладача проксі-сертифікат, перейменувати у «cert.bin» назначити йому права доступу 400 та покласти в каталог /home/<username>/.globus/

7) Запустити задачу на виконання.
Приклад задачі hello-world на мові Bash (job.sh):

```
#!/bin/sh
echo "Hello grid!"
hostname
date
pwd
```

Приклад файла-специфікації у xrsl-форматі (job.xrsl):

```
&
(executable=job.sh)
(inputFiles=(job.sh ""))
(stdout="out.txt")
(stderr="err.txt")
(outputFiles=("out.txt" "")("err.txt" ""))
(jobname="Hello grid")
```

Виконання задачі. У разі успішного виконання команди буде отриманий унікальний ідентифікатор задачі.

```
$ arcsub jib.xrsl -c grid.isma.kharkov.ua
```

Перевірка статусу виконання задачі за її ідентифікатором:

```
$ arcstat
https://grid.isma.kharkov.ua:60443/arex/9PCLDmmJ79rnR1181mI4zItmABFK
DmABFKDmMlFKDm8UFKDmjfyq7m
```

Отримання результатів виконання задачі (для статусу FINISHED):

```
    $ arcget
https://grid.isma.kharkov.ua:60443/arex/9PCLDmmJ79rnR1181mI4zItmABFK
DmABFKDmMlFKDm8UFKDmjfyq7m
```

1.4 Завдання для самостійної роботи

1) Запустити обчислювальну задачу на іншому ресурсі. Порівняти час виконання задачі та пояснити, чим може бути обумовлена різниця у швидкості виконанні задачі.

2) Модифікувати задачу таким чином, щоб вона завершувалась помилкою в виводом у стандартний потік помилок. Отримати результати виконання задачі за допомогою команди arcget. Пояснити результат.

2 Лабораторна робота №2 Розгортання розподіленого середовища на базі Apache Hadoop

2.1 Мета роботи

Ознайомитись з існуючими реалізаціями розподілених середовищ (Apache Hadoop). Навчитись розгортати та використовувати розподілені середовища для вирішення ресурсномістких задач

2.2 Теоретичні відомості

Apache Hadoop – проект, який колись був частиною відкритої пошукової системи Nutch, але пізніше відокремився від неї як реалізація Distributed File System і парадигми MapReduce.

Hadoop складається з двох модулів. Це реалізація парадигми розподіленої файлової системи (яка називається HDFS) і MapReduce, тобто реалізація MapReduce-фреймворка.

Distributed file system – розподілена файлова система. Основні вимоги:

- зберігання великих обсягів даних;
- прозорість, работа як зі звичайною файловою системою – з можливістю відкривати файли і не думати про реалізацію розподіленого зберігання;
- масштабованість – можливість додавати нові ноди для зберігання даних;
- надійність – вихід з ладу деякого відсотка вузлів не повинно зашкодити цілісності всієї системи.

DFS використовує кластер, що складається з керуючої master-ноди, і slaves, на яких зберігаються дані. На мастер-ноді зберігається таблиця файлів. Кожен файл розбитий на блоки. На мастері зберігається інформація, на яких конкретно кластерних машинах знаходяться блоки файлів.

При читанні/запису даних майстер надає інформацію про розташування блоків файлу: на якій машині зберігаються конкретні блоки та їх порядковий номер.

Для забезпечення надійності, кожен блок зберігається в декількох примірниках, на декількох машинах.

MapReduce – концепція розподілу обчислень.

Робота MapReduce складається з двох кроків: Map і Reduce.

На першому кроці (MAP) відбувається попередня обробка вхідних даних. Для цього один з комп'ютерів (master node) отримує вхідні дані задачі, розділяє їх на частини і передає іншим комп'ютерам (worker node).

На другому кроці (REDUCE) відбувається агрегування попередньо оброблених даних на кроці MAP. Головний вузол отримує відповіді від робочих вузлів і на їх основі формує результат – рішення задачі.

2.3 Порядок виконання роботи

Налаштування кластеру з одним вузлом.

1) Оновлення індексу репозиторіїв:

```
$ sudo apt-get update
```

2) Встановлення Java:

```
$ sudo apt-get install default-jdk
$ update-alternatives --config java
```

Версія Java повинна бути не меньше 1.7. Перевірка:

```
$ java –version
```

3) Встановлення SSH:

```
$ sudo apt-get install ssh
```

4) Встановлення RSYNC:

```
$ sudo apt-get install rsync
```

5) Генерація ключів для SSH:

```
$ ssh-keygen -t dsa -P ' ' -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

6) Завантажуємо Hadoop (TAR.GZ архів) з офіційного сайту.

7) Розпакування Hadoop:

```
$ sudo tar -zxvf hadoop-<version>.tar.gz
```

8) Додавання змінних середовища в ~/.bashrc

```
#Hadoop Variables
export JAVA_HOME=<шлях до каталогу JAVA>
export HADOOP_HOME=<шлях до розпакованого каталогу Hadoop>
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

9) Оновлення конфігурації після внесення змін:

```
$ source ~/.bashrc
```

10) В каталозі hadoop відредагувати hadoop-env.sh

```
export JAVA_HOME=<шлях до каталогу Java>
```

11) Редагування core-site.xml

```
 <configuration>
  <property>
      <name>fs.defaultFS</name>
      <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

12) Редагування yarn-site.xml

```
<configuration>
 <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
 </property>
 <property>
      <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>
      <value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
</configuration>
```

13) Копіювання шаблону:

```
$ sudo cp mapred.site.xml.template mapred-site.xml
```

14) Редагування шаблону (mapred-site.xml):

```
 <configuration>
      <property>
           <name>mapreduce.framework.name</name>
           <value>yarn</value>
      </property>
 </configuration>
```

15) Редагування hdfs-site.xml

```
 <configuration>
      <property>
           <name>dfs.replication</name>
           <value>1</value>
      </property>
      <property>
           <name>dfs.namenode.name.dir</name>

 <value>
 file:<каталог Hadoop>/hadoop_data/hdfs/namenode
 </value>
      </property>
      <property>
           <name>dfs.datanode.data.dir</name>
           <value>
```

```
file: <каталог Hadoop>/hadoop_store/hdfs/datanode
</value>
     </property>
</configuration>
```

16) Створення файлів нод:

```
$ mkdir -p <каталог Hadoop>/hadoop_data/hdfs/namenode
$ mkdir -p <каталог Hadoop>/hadoop_data/hdfs/datanode
```

17) Зміна поточного користувача та групи каталогу з Hadoop:

```
$ sudo chown <користувач>:<група> -R <каталог Hadoop>
```

18) Форматування та запуск

```
$ hdfs namenode -format
$ start-all.sh
```

Перевірка статусу:

```
$ jps
```

2.4 Завдання для самостійної роботи

Розробити та виконати задачу у розподіленому середовищі Hadoop.
Варіанти завдань:

1. Множення квадратної матриці розмірності N на вектор розмірністю N.
2. Розкладання числа N на прості множники.
3. Пошук мінімального елемента в матриці розмірності N.
4. Пошук максимального елемента в матриці розмірності N.
5. Сортування масиву розмірністю N в прямому порядку.
6. Сортування масиву розмірністю N в оберненому порядку.
7. Сортування матриці розмірністю N по горизонталі зліва направо.
8. Сортування матриці розмірністю N по горизонталі справа-наліво.
9. Сортування матриці розмірністю N по вертикалі зверху-вниз.Сортування матриці розмірністю N по горизонталі знизу-вверх.
10. Множення матриці розмірністю NxM на матрицю розмірністю MxK.
11. Пошук елементу с заданим значенням в матриці розмірністю N.

3 Лабораторна робота №3 Розподілені системи зберігання та обробки даних

3.1 Мета роботи

Ознайомитись з noSQL-системами зберігання та обробки даних.

3.2 Теоретичні відомості

Бази даних NoSQL – це нереляційні бази даних з можливістю масштабування, оптимізовані для використання моделей даних без схем. Бази даних NoSQL набули широкого поширення, оскільки вони спрощують розробку, забезпечують відмовостійкість і низьку затримку запитів. Такі бази даних можуть використовувати різні моделі даних, включаючи стовпчасті, документні, графові дані і сховища пар «ключ-значення» в пам'яті.

Для баз даних NoSQL схема, як правило, не використовується. Для отримання конкретних значень, наборів стовпців, частково структурованих файлів JSON, XML або інших документів, що містять відповідні атрибути елементів, зазвичай використовується ключ секції.

Бази даних NoSQL часто нехтують деякими властивостями ACID традиційних СУБД для забезпечення більш гнучкої моделі даних з можливістю горизонтального масштабування. Вони розраховані на горизонтальне масштабування з використанням розподілених кластерів недорогого апаратного забезпечення для підвищення продуктивності без затримок.

3.3 Порядок виконання роботи

1. Ознайомитись з документацією MongoDB та RethinkDB:
https://www.mongodb.com/
https://www.rethinkdb.com/

2. Пояснити особливості цих систем

3. Встановити RethinkDB. Приклад встановлення для ОС Ubuntu:
3.1 Реєстрація репозиторію RethinkDB та ключа до нього:

```
    # export DISTRIB_CODENAME=<назва дистрибутиву Ubuntu>
    # source /etc/lsb-release && echo "deb
http://download.rethinkdb.com/apt $DISTRIB_CODENAME main" | sudo tee
/etc/apt/sources.list.d/rethinkdb.list

    # wget -qO- https://download.rethinkdb.com/apt/pubkey.gpg | sudo
apt-key add -
```

3.2 Встановлення програмного забезпечення:

```
    $ sudo apt-get update
    $ sudo apt-get install rethinkdb
```

3.3 Також за бажанням можна скомпілювати з вихідних текстів:
- встановлення допоміжних бібліотек

```
$ sudo apt-get install build-essential protobuf-compiler python libprotobuf-dev libcurl4-openssl-dev libboost-all-dev libncurses5-dev libjemalloc-dev m4
```

- завантаження архіву з вихідними текстами:

```
$ wget https://download.rethinkdb.com/dist/rethinkdb-2.3.6.tgz
```

- розпакування архіву

```
$ tar xf rethinkdb-2.3.6.tgz
```

- компіляція та встановлення:

```
$ cd rethinkdb-2.3.6
$ ./configure --allow-fetch
$ make
$ sudo make install
```

4. Далі необхідно встановити клієнтські драйвери для мови програмування, з якою ви працюєте. Офіційно підтримуються Java, JavaScript, Ruby та Python. Також доступні драйвери і для інших мов, що підтримуються іншими розробниками. Повний список є тут:
https://www.rethinkdb.com/docs/install-drivers/

5. Приклад встановлення клієнтського драйверу для Java.
Для системи Maven:

```
<dependencies>
  <dependency>
    <groupId>com.rethinkdb</groupId>
    <artifactId>rethinkdb-driver</artifactId>
    <version>2.3.3</version>
  </dependency>
</dependencies>
```

Для системи Gradle:

```
dependencies {
    compile group: 'com.rethinkdb', name: 'rethinkdb-driver', version: '2.3.3'
}
```

Також можливо просто завантажити JAR-файл з усіма необхідними залежностями, але такий спосіб не рекомендується.

6. Запуск сервера RethinkDB (якщо він не запустився автоматично):

```
$ rethinkdb
```

7. Приклад підключення до бази даних:

```
import com.rethinkdb.RethinkDB;
import com.rethinkdb.gen.exc.ReqlError;
import com.rethinkdb.gen.exc.ReqlQueryLogicError;
import com.rethinkdb.model.MapObject;
import com.rethinkdb.net.Connection;

public static class MainApp {

    public static void main(String[] args) {
        Connection conn = RethinkDB.r
            .connection()
            .hostname("localhost")
            .port(28015)
            .connect();
    }

}
```

8. Документацію по RethinkDB можна подивитись тут: https://www.rethinkdb.com/docs/guide/java/

9. Масштабування

RethinkDB надає функції для шардингу та реплікації бази даних. Для цього використовується веб-інтерфейс або ж команди ReQL.

В web UI (localhost:8080), можна просто зазничити кількість шард. RethinkDB автоматично визначить, як треба розділити дані, щоб підтримувати баланс між шардами.

Порядок дій:

- Перейти по потрібної таблиці Tables -> [table name].
- Натиснути кнопку "Reconfigure".
- Встановити кількість шард та реплік.
- Натиснути кнопку "Apply".

10. Через web UI можна конфігурувати сервери з тегом default.

Під час запуску rethinkdb-сервера можна вказати теги, до яких він належить. Всі сервери групуються в кластери по тегам.

```
$ rethinkdb --server-tag <tag1> --server-tag <tag2>
```

Також при додаванні серверу до кластеру необхідно відредагувати конфігураційний файл /etc/rethinkdb/instances.d/cluster_instance.conf, вказа-

вши у ньому параметр join з відкритою адресою кластера. За замовчуванням не можна підключитись до instance сервера зовні. Тому для цього необхідно поправити конфігураційний файл cluster_instance.conf master-ноди, змінивши параметр bind=all.

11. Запуск проксі-нод.

Проксі-ноди не зберігають дані. Вони лише виконують функцію маршрутизації запитів у кластері для більш ефективного розподілення ресурсів. Приклад запуску проксі-ноди:

```
$ rethinkdb proxy --join <hostname>:<port>
```

3.4 Завдання для самостійної роботи

1. Розгорнути кластер RethinkDb з 3 нод, 2 з яких знаходяться на віртуальних машинах. Налаштувати шардинг та реплікацію між нодами кластера.

2. Заповнити базу довільними даними на 100 Мб, 1 Гб та 10 Гб та дослідити, як розподіляються дані між машинами.

4 Лабораторная работа №4 Керування віртуальними оточеннями розподілених ресурсів на базі Open Nebula

4.1 Мета роботи

Ознайомитись з програмною платформою OpenNebula

4.2 Теоретичні відомості

OpenNebula – програмна платформа, призначена для організації управління хмарною інфраструктурою і віртуальними оточеннями. Платформа дозволяє організувати функціонування розподіленої інфраструктури для динамічно розгорнутих багаторівневих сервісів (груп взаємопов'язаних віртуальних машин), комбінуючи ресурси локального дата-центру і зовнішніх хмарних провайдерів. Зокрема, OpenNebula дозволяє на своїх потужностях підняти інфраструктуру для надання сервісів IaaS, схожу на Amazon EC2. В наявності є засоби для організації розгортання віртуальних оточень, моніторингу, контролю доступу, забезпечення безпеки та управління сховищем.

Вихідний код системи повністю відкритий під ліцензією Apache. Пакети доступні для Ubuntu, openSUSE, RHEL/CentOS і Debian. Серед найвідоміших компаній та організацій, що використовують OpenNebula, можна відзначити: CERN, Європейське космічне агентство, FermiLab і China Mobile.

Більш докладний посібник з встановлення та налаштування openNebula можна знайти на офіційному сайті розробника http://opennebula.org.

4.3 Порядок виконання роботи

Рекомендується встановлювати OpenNebula на VMWare.

1) Встановлення ноди Open Nebula на frontend-машині. Frontend-машина буде використовуватись як інтерфейс доступу до ресурсів.

Перед тим як починати установку, потрібно переконатися, що в налаштуваннях даної віртуальної машини встановлений чекбокс Virtualize Intel VT-x/EPT or AMD-V/RVI.

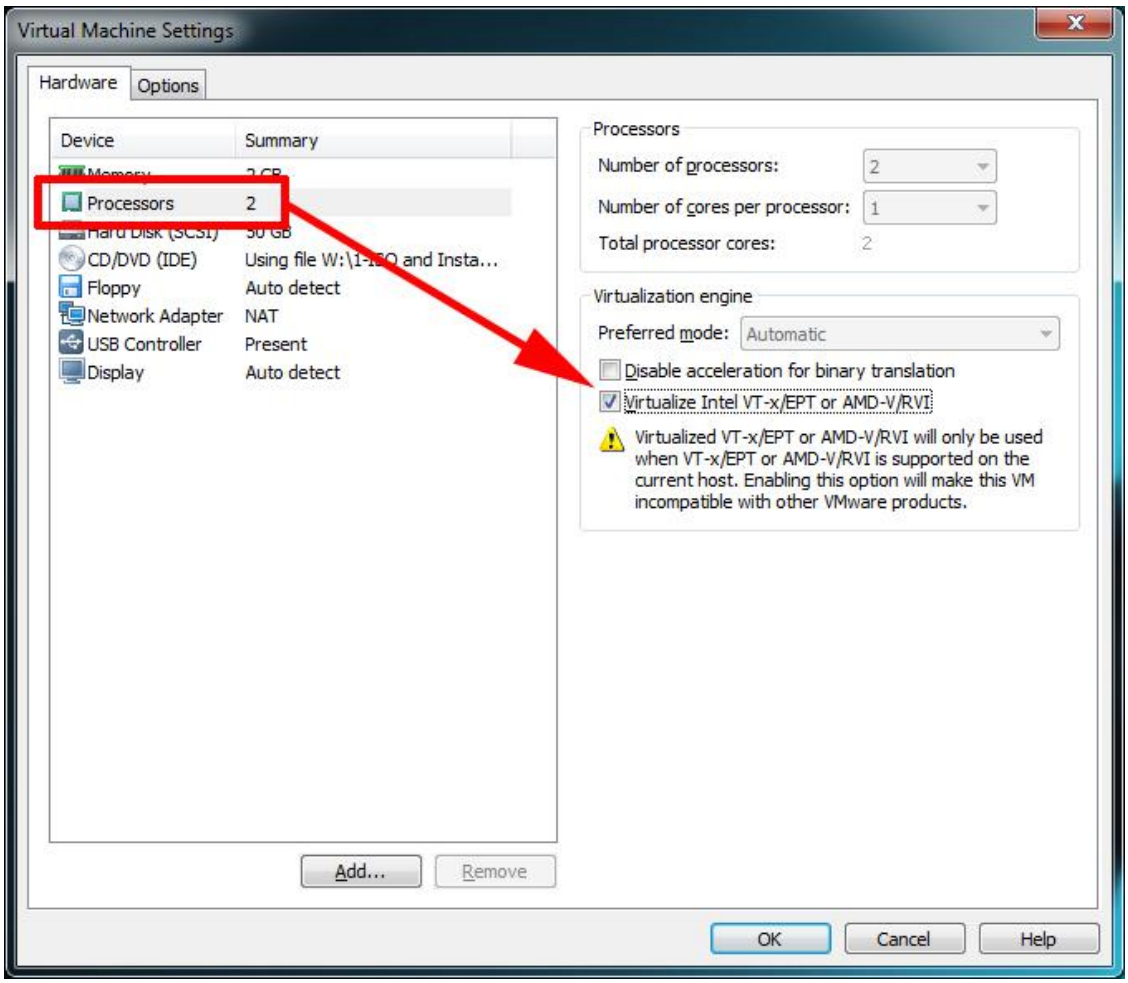

Рисунок 4.1 – Перевірка параметру Virtualize Intel VT-x/EPT or AMD-V/RVI

Також про всяк випадок потрібно перевірити що в BIOS включена підтримка даної функції. Для процесорів Intel необхідно знайти властивість Intel® Virtualization Technology і встановити його в Enable. Після чого можна перейти до процесу установки OS на віртуальну машину.

Після завершення установки необхідно переконатися, що все правильно встановлено. Виконуємо перевірку що включена підтримка VT-x, для цього виконуємо в терміналі команду:

```
grep -E 'svm|vmx' /proc/cpuinfo
```

Додаємо openNebula репозиторій:

```
# wget -q -O-
http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key add -

# echo "deb
http://downloads.opennebula.org/repo/4.12/Ubuntu/14.04/ stable
opennebula" /etc/apt/sources.list.d/opennebula.list
```

Встановлюємо необхідні пакети:

```
# apt-get update

# apt-get install opennebula opennebula-sunstone nfs-kernel-
server libvirt-dev
```

Після установки даних пакетів додаємо поточного користувача в групу libvirt, а також необхідно впевнитися, що запущено два нових процеси:

oned – демон openNebula

sunstore – графічний користувальницький інтерфейс.

Потрібно налаштувати інтерфейс доступу, так як він працює після стандартної установки тільки на localhost. Для цього необхідно змінити вміст файлу /etc/one/sunstone-server.conf, параметр host: 127.0.0.1 треба змінити на host: 0.0.0.0, та перезапустити демон:

```
# /etc/init.d/opennebula-sunstone restart
```

Після цього можна потрапити в графічний інтерфейс адміністратора openNebula набравши в браузері рядок:

```
http:// <ip_address_vm>:9869
```

Для входу в систему треба ввести імя користувача в поле Login та пароль opennebula.

Після цього ми буде отриманий доступ до графічного інтерфейсу openNebula.

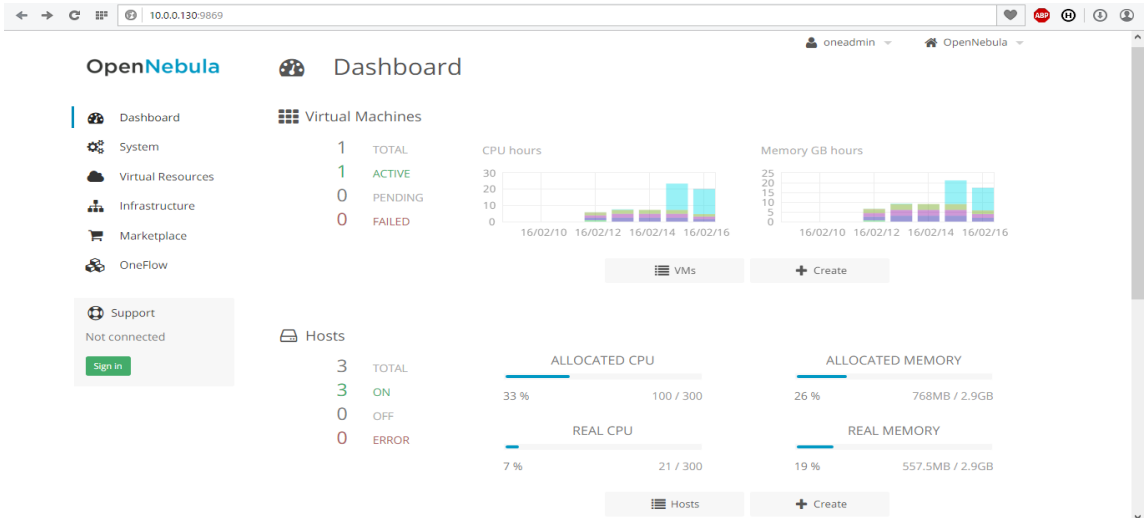

Рисунок 4.2 – Інтерфейс користувача OpenNebula

Для більшої зручності щоб не запускати кожен раз демон opennebula-sunstone при перезавантаженні або включенні машини, його можна додати в автозапуск.

Далі потрібно провести установку на інші ноди. На кожну ноду встановлюється OS з такою ж конфігурацією, як і на Front-end машині. Після установки OS також рекомендується перевірити наявність флагу svm|vmx в /proc/cpuinfo.

Далі описаний процес установки openNebula на одну ноду, на інші ноди процес установки аналогічний (або можна склонувати ноду, тільки потім налаштувати мережу щоб у клонованій машини був інший IP).

Додаємо openNebula репозиторій:

```
# wget -q -O-
http://downloads.opennebula.org/repo/Ubuntu/repo.key | apt-key add -
```

```
# echo "deb
http://downloads.opennebula.org/repo/4.12/Ubuntu/14.04/ stable
opennebula" /etc/apt/sources.list.d/opennebula.list
```

Встановлюємо пакети:

```
# apt-get update
```

```
# apt-get install opennebula-node nfs-common bridge-utils
```

Налаштовуємо мережу. Необхідно щоб головний мережевий інтерфейс був приєднаний до мосту. Якщо конфігурацію мережевого інтерфейсу ми отримуємо через DHCP то для інтерфейсу, наприклад eth0,можна налаштувати наступну конфігурацію:

Файл /etc/network/interfaces

```
auto lo
iface lo inet loopback
auto br0
iface br0 inet dhcp
bridge_ports eth0
bridge_fd 9
bridge_hello 2
bridge_maxage 12
bridge_stp off
```

Якщо ж конфігурацію мережевого інтерфейсу задається статично, то необхідно внести наступні зміни:

Файл /etc/network/interfaces

```
auto lo
iface lo inet loopback
auto br0
iface br0 inet static
address 192.168.0.10
network 192.168.0.0
netmask 255.255.255.0
broadcast 192.168.0.255
gateway 192.168.0.1
bridge_ports eth0
bridge_fd 9
bridge_hello 2
bridge_maxage 12
bridge_stp off
```

Після чого перезапускаємо мережевий демон:

```
# /etc/init.d/networking restart
```

Налаштовуємо NFS:

Додаємо в файл /etc/fstab наступне:

```
<ip_address_manager_node>:/var/lib/one/ /var /lib/one/nfs
soft, intr, rsize=8192, wsize=8192, noauto
```

Після чого монтуємо NFS:

```
# mount /var/lib/one/
```

Тепер необхідно налаштувати Qemu щоб поточний користувач міг керувати libvirt як root:

Файл /etc/libvirt/qemu.conf

```
user = "ім'я користувача"
group = "група (зазвичай = імені)"
dynamic_ownership = 0
```

Після цього потрібно перезапустити libvirt для застосування цих змін:

```
# service libvirt-bin restart
```

На цьому налаштування ноди завершено, залишається тільки приєднати її до менеджера (Front-end машини). Для цього переходимо в командний рядок front-end, переходимо на користувача oneadmin:

```
# su oneadmin
```

Налаштовуємо ssh-доступ для того щоб дані між VM ходили без запиту ключа, в мережі можна знайти багато статей як налаштувати ssh (команди ssh-keygen, ssh-copy-id), додаємо хости:

```
$ onehost create <ip_manager> -i kvm -v kvm -n dummy
$ onehost create <ip_node> -i kvm -v kvm -n dummy
```

Щоб зрозуміти, що тут відбувається, рекомендується почитати man-сторінку з onehost. На цьому установка openNebula закінчується. В результаті маємо CLI і GUI, які дуже зрозумілі і легкі у використанні. Щоб подивитися, якими командами можна управляти openNebula через CLI, треба перейти до офіційної документації:

http://docs.opennebula.org/pdf/4.4/opennebula_4.4_administration_guide.pdf.

При управлінні за допомогою GUI в різних вкладках можна налаштовувати під себе openNebula.

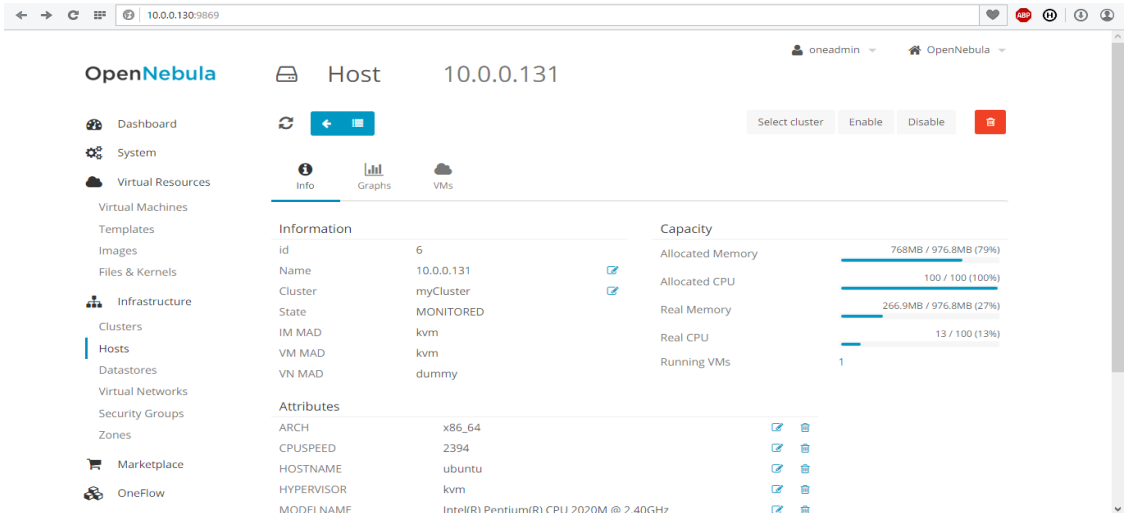

Рисунок 4.3 – Навантаження на ресурси OpenNebula

4.4 Завдання для самостійної роботи

1. Розгорнути OpenNebula з 4-ма віртуальними машинами ОС Linux без графічної оболонки (достатньо 512 Мб памяті для кожної із таких машин). Одна віртуальна машина – frontent-машина, що надає доступ до веб-інтерфейсу OpenNebula. Інші машини – ноди OpenNebula.

2. Графічний інтерфейс має бути доступним як із хост-машини, на якій запущені віртуальні машини, так і з інших машин, що знаходяться в цій же локальній мережі.

Рекомендована література

1. Andrew S. Tanenbaum, Maarten van Steen. Distributed Systems: Principles and Paradigms. – Upper Saddle RIiver, NJ: Pearson Prentice Hall, 2nd edition, 2006. – 686 p.
2. Nancy A. Lynch. Distributed Algorithms. – San Francisco, Calif.: Morgan Kaufmann Publishers, 1996. – 872 p.
3. Ajay D. Kshemkalyani, Mukesh Singhal. Distributed Computing: Principles, Algorithms and Systems. – Cambridge University Press, 2008. – 736 p.
4. Mikito Takada. Distributed Systems for Fun and Profit (Електронний ресурс http://book.mixu.net/distsys/ebook.html).
5. Apache Hadoop (Електронний ресурс http://hadoop.apache.org/).
6. Redis (Електронний ресурс https://redis.io/)
7. MongoDB for Giant Ideas (Електронний ресурс https://www.mongodb.com/).
8. Torque Resource Manager (Електронний ресурс http://www.adaptivecomputing.com/products/open-source/torque/).
9. Consensus Protocols: Paxos (Електронний ресурс http://the-paper-trail.org/blog/consensus-protocols-paxos/).