УДК 681.3

Деревянченко О. В., к.ф.-м. н., доц.,
Хавро А.Ю., студент

Застосування системи ПАРКС.NET та Amazon EC2 для хмарних обчислень

Київський національний університет імені Тараса Шевченка, 83000, м. Київ, пр-т. Глушкова 4д, e-mail: alexanderder@mail.ru

O.V. Derevianchenko, PhD, Associate Professor
A.U. Khavro, student

The applying of PARCS.NET system and Amazon EC2 for cloud computing

Taras Shevchenko National University of Kyiv, 83000, Kyiv, Glushkova st., 4d,
e-mail: alexanderder@mail.ru

В статті запропоновано застосування системи для паралельних розподілених обчислень ПАРКС.NET (ПАРКС - Паралельні Асинхронні Рекурсивні Керуючі Системи) та хмарних обчислень. У якості технології для хмарних обчислень було використано Amazon Elastic Compute Cloud (Amazon EC2).

В роботі надано опис роботи систем ПАРКС.NET та можливості її налаштування для роботи з технологією Amazon Elastic Compute Cloud. Наведено результати тестування систем для вирішення класичної задачі множення матриць. Розглянуто можливості цих систем при вирішенні практичних задач паралельного програмування.

Ключові слова: хмарні обчислення, паралельні розподілені обчислення, система ПАРКС, C#, Amazon EC2.

In the article provides the use of the system for parallel distributed computing PARCS.NET (PARCS - Parallel Asynchronous Recursive Control System) and cloud computing. As the technology for cloud computing was used Amazon Elastic Compute Cloud (Amazon EC2).

The paper provided a description of the systems PARCS.NET and its possible settings for the technology Amazon Elastic Compute Cloud.

The basic terms of the PARCS-technology are point, programming channel and algorithmic module. Pattern of control space is a graph. Top is a point of control space and edge is programming channels. The points are connected per programming channels. Each point of control space is assigned with the algorithmic module. The algorithmic module is a procedure PARCS and extension of the basic programming languages. Algorithmic modules are executed in parallel. In practice this means that in a computer network one major computer (Server) is selected. This computer is accessible to all others (Clients). It generates a queue of all tasks. Tasks from the queue are sent for computing. Each computer obtains a task from the queue and executes it. After finishing computing the active task it sends the results to the server and selects the new task. The model is designed for distributed network. Description of these systems is provided

The results of testing systems for solving the classical problem of matrix multiplication are presented. Capabilities of these systems in solving practical problems of parallel programming were considered.

Keywords: cloud computing, parallel distributed computing, PARCS system, C#, Amazon EC2.

Статтю представив д.ф.-м.н., проф. Анісімов А.В.

Вступ

В даній статті розглядається застосування технології для паралельних розподілених обчислень ПАРКС (Паралельні Асинхронні Рекурсивні Керуючі Системи) [1], а саме системи ПАРКС.NET [6,7] та хмарних обчислень. У якості технології для хмарних обчислень було використано Amazon Elastic Compute Cloud (Amazon EC2) [8]. Ця робота присвячена розширенню (паралельні властивості) множини алгоритмічних мов програмування (Pascal, C, JAVA[2-4]), які вже реалізовані за допомоги технології ПАРКС, новими за допомогою технології .NET та мови програмування C# [5], а також використанню таких технологій у хмарних обчисленнях.

Технологія ПАРКС

ПАРКС (паралельні асинхронні рекурсивні керовані системи) - технологія програмування, що являє собою деяку сукупність програмних засобів, які забезпечують процес розробки і реалізації алгоритмів паралельної обробки інформації. Вона базується на концепції *керуючого простору* (КП) [1] – динамічний граф, який використовується для опису логічної та комунікаційної структури досліджуваної задачі.

Основні терміні ПАРКС-технології:

1) точка;
2) програмний канал (ПК);
3) алгоритмічний модуль (АМ).

Структура КП – граф, вершини якого – точки КП, ребра – ПК. До кожної точки КП приписаний АМ, який є процедурою ПАРКС-розширення базової мови. В нашому випадку ми застосовуємо базову мову програмування C#.

Програмні засоби розширення базових алгоритмічних мов програмування[1]:

1. Програмні засоби, що забезпечують побудову і модифікацію КП.
2. Програмні засоби, що забезпечують зберігання та передачу інформації за допомогою КП.
3. Програмні засоби, що забезпечують роботу з АМ.
4. Програмні засоби, що забезпечують процедурно – об'єктно-орієнтоване програмування.

Технологія Microsoft .NET Framework

Microsoft .NET Framework – сучасна програмна платформа, що дозволяє створювати програми із застосуванням комп'ютерних мереж та паралельних обчислень, а також надає можливість переносу програм між різними апаратними платформами, на яких встановлене середовище Microsoft .NET Framework. Однією з переваг .NET є сумісність бібліотек, написаних різними мовами. Система ПАРКС-.NET написана на мові C# [5], але її можна використовувати і для інших мов .NET, наприклад Visual Basic або F#.

До позитивних якостей технології .NET, що вплинули на її вибір для проектування системи ПАРКС.NET, також належать:

- наявність мови C#, яка набуває все більшої популярності завдяки простій об'єктній моделі та постійному розвитку;
- відповідність сучасним вимогам: бібліотека TPL (Task Parallel Library), яка використовується в системі ПАРКС.NET і надає широкий набір можливостей для розробки паралельних програм, підтримка засобів мережевого програмування, забезпечення пересилання даних по комп'ютерній мережі, рефлексія – процес виявлення типів під час виконання та інше.

Система ПАРКС.NET

Архітектура системи складається з наступних частин [6,7]:

1. *Parcs* – основна бібліотека класів системи. Містить всі основні класи, що будуть використовуватись для моделювання паралельних обчислень.
2. *Daemon* – клієнт. Являє собою програму, за допомогою якої будуть проводитись обчислення.
3. *HostServer* – сервер. Має список доступних клієнтів, а також інформацію про поточні задачі та точки. При запуску модуль починає роздавати завдання клієнтам в залежності від кількості ядер процесорів (потужності) та їх завантаженості.
4. АМ – окремий проект, що використовує бібліотеку *Parcs* та реалізує інтерфейс *IModule*.

На всіх машинах, що призначені для обчислень, запускається програма *Daemon*, а початковий АМ та *HostServer* можуть запускатися або на тих же, або на інших машинах. В окремому випадку навіть всі три елементи системи можуть бути запущені в одному місці. Файл server.txt використовується початковим АМ для знаходження комп'ютера, на якому запущений *HostServer* (IP-адреса).

У створеній реалізації клієнта та сервера зв'язок проходить у такій послідовності:

1.При запуску модуля сервер перевіряє всі зазначені у файлі hosts.txt і видаляє ті, з якими не вдалося з'єднатися. Після цього сервер запитує у кожного з клієнтів кількість ядер процесора і зберігає собі ці результати.

2.Сервер розподіляє точки, що створюються в алгоритмічному модулі, по

комп'ютерах, на яких виконується програма *Daemon*.

3.Після цього на комп'ютері, де буде створена нова точка, початковий АМ передає exe-файл, в якому міститься інформація про всі АМ, що будуть запущені під час виконання поточної задачі. За допомогою рефлексії *Daemon* створює екземпляр класу АМ і викликає метод Run(). Таким чином запускаються необхідні обчислення.

4.Після завершення обчислень *Daemon* відсилає результат назад на початковий АМ. Після того, як завершилися обчислення на всіх машинах, початковий АМ оброблює всі результати, виводить загальний результат на екран та зберігає його в файл.

Amazon Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) - веб-сервіс [8], який надає хмарні обчислювальні потужності. Сервіс входить в інфраструктуру Amazon Web Services (AWS).

Простий веб-інтерфейс сервісу дозволяє отримати доступ до обчислювальних потужностей і налаштувати ресурси з мінімальними витратами. Він надає користувачам повний контроль над обчислювальними ресурсами, а також доступне середовище для роботи. Служба скорочує час, необхідний для отримання і завантаження нового сервера.

За допомогою EC2 дозволяється створювати хмарні віртуальні машини, використовуючи потужності хмари Amazon. Користувачу надається вибір операційної системи: Windows Server або різні версії Linux. Також можна вибрати характеристики віртуальної машини.

Серед інших особливостей EC2 варто відзначити можливість створення образів віртуальних машин (Amazon Machine Image (AMI)), а також їх зберігання. AMI дозволяє встановлювати програми, бібліотеки, дані та налаштовувати пов'язані з ними параметри. За допомогою AMI можна створювати нові машини на основі існуючих (фактично копіювати віртуальні машини).

Також важливою особливістю є можливість налаштування безпеки та мережевого доступу. Це дозволяє відкрити доступ до віртуальної машини для зовнішнього світу (за замовчуванням він закритий), а також дозволити комунікацію між самими віртуальними машинами.

Веб-інтерфейс *Amazon Elastic Compute Cloud* легко дозволяє запускати і зупиняти віртуальні машини, а також робити всі вищеописані дії.

Використання технології Amazon Elastic Compute Cloud

Для використання *Amazon Elastic Compute Cloud* необхідно зареєструватися на сайті http://aws.amazon.com/. Для цього потрібно ввести свої персональні дані і пройти підтвердження реєстрації по телефону. Також при реєстрації потрібно вказати номер кредитної картки, хоча сервіс можна використовувати і безкоштовно – кожному користувачу надається обмежений пакет користування сервісом на рік.

Після проходження реєстрації користувач може починати працювати з сервісом. Так на Рис.1 можна побачити веб-інтерфейс сервісу *Amazon Elastic Compute Cloud*. Видно, що на даний момент створено одну віртуальну машину з ім'ям «host server» і приватним IP. Варто зазначити, що при перезапуску машин приватний IP залишається сталим, а публічний IP може змінюватися. На рисунку видно, що машині не присвоєно публічний IP, оскільки вона знаходиться у зупиненому стані.

Для того, щоб створити нову віртуальну машину, потрібно натиснути на кнопку "Launch Instance". Після цього потрібно обрати образ (AMI), з якої буде створена віртуальна машина (VM), Рис.2. Це може бути як стандартний образ операційної системи, такі як Red Hat Enterprise Linux або Microsoft Windows Server 2012, так і власноруч створені образи, або будь-які образи, створені спільнотою.

Наступним кроком є вибір типу машини. Є один безкоштовний варіант: машина з одним процесором Intel Xeon з потужністю 2.5 GHz і 1 ГБ оперативної пам'яті Рис.3.

Далі можна обрати інші налаштування машини. Зокрема, можна створити декілька віртуальних машин за один раз Рис.4.

Далі можна додати місце на диску (за замовчуванням 30 ГБ), позначити машину тегами, налаштувати security group (множина правил мережевого екрану). Ці кроки можна пропустити, але для роботи системи ПАРКС.NET необхідно забезпечити зв'язок між машинами, тому покажемо, як можна налаштувати мережевий екран.

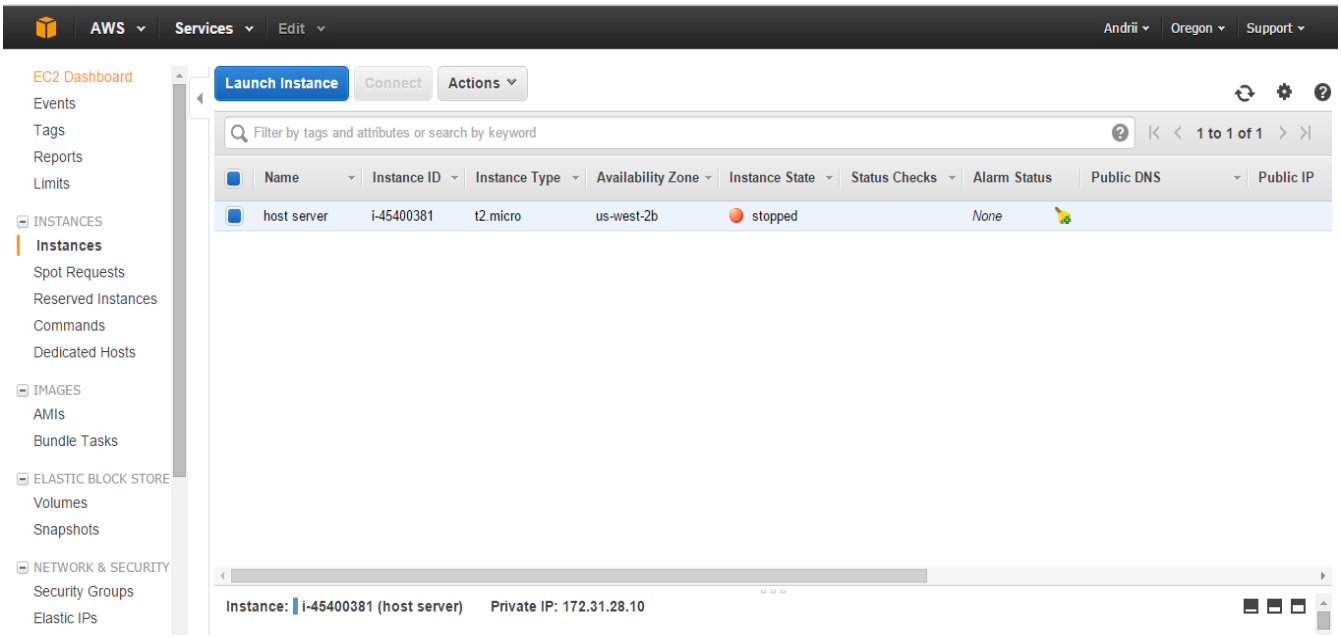

Рис.1. Сервіс *Amazon EC2*

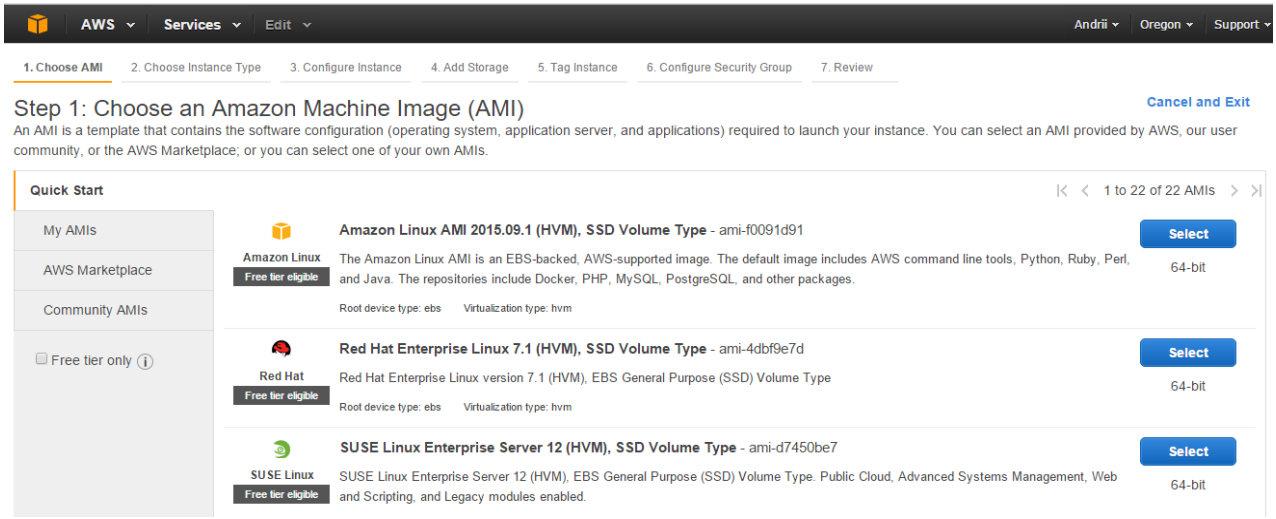

Рис.2. Створення нової VM

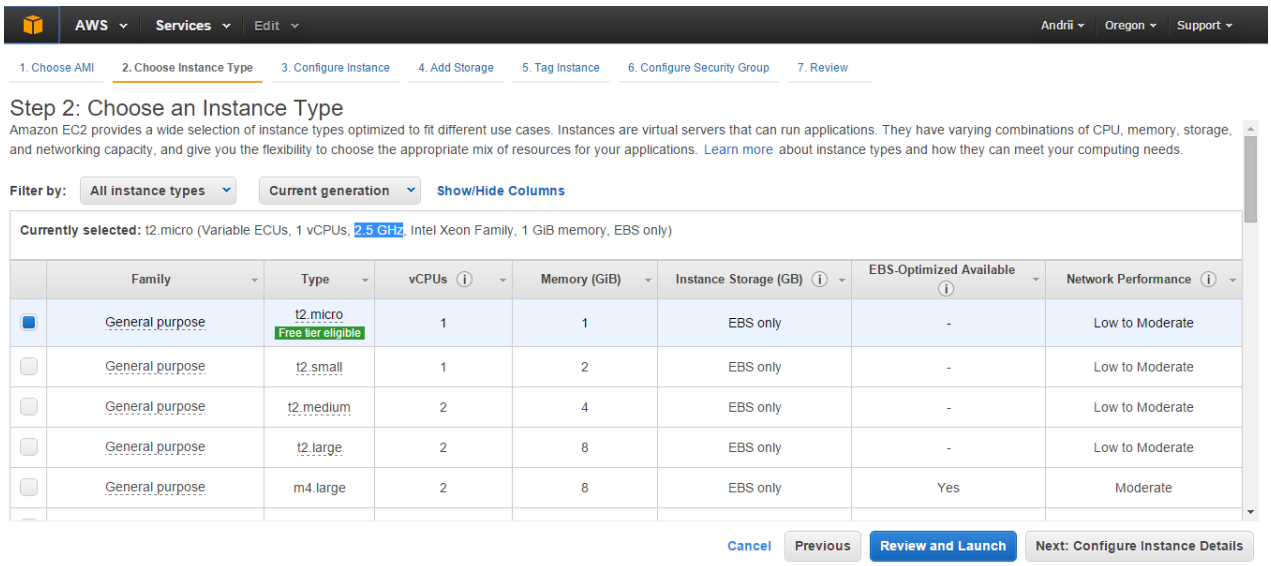

Рис.3. Вибір типу машини

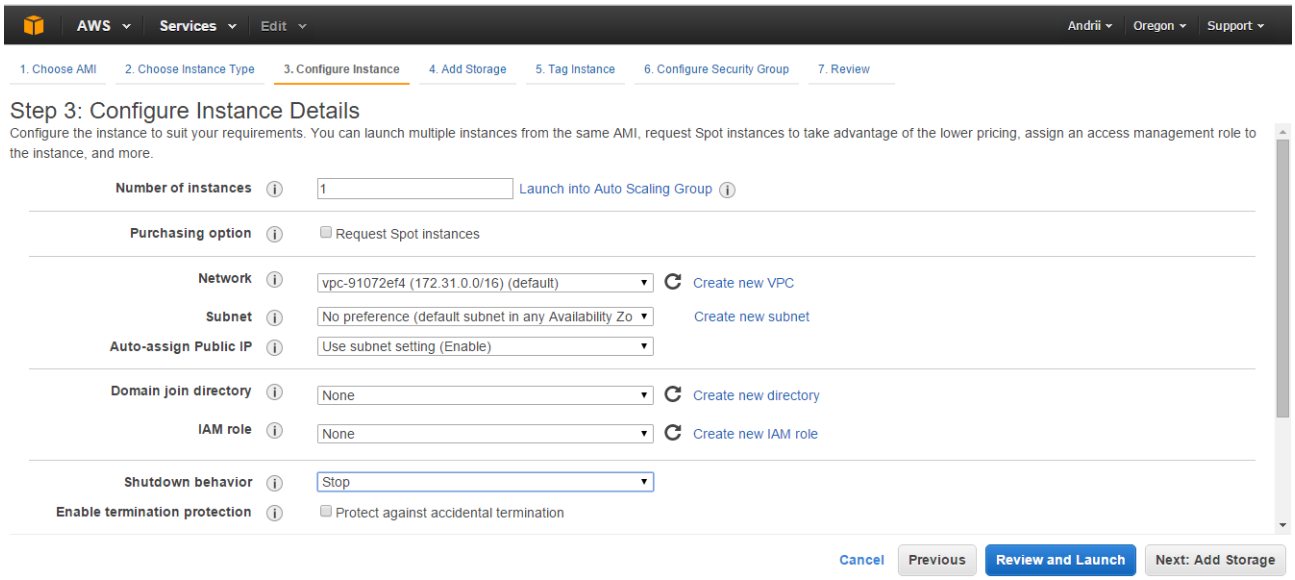

Рис.4. Створення віртуального простору

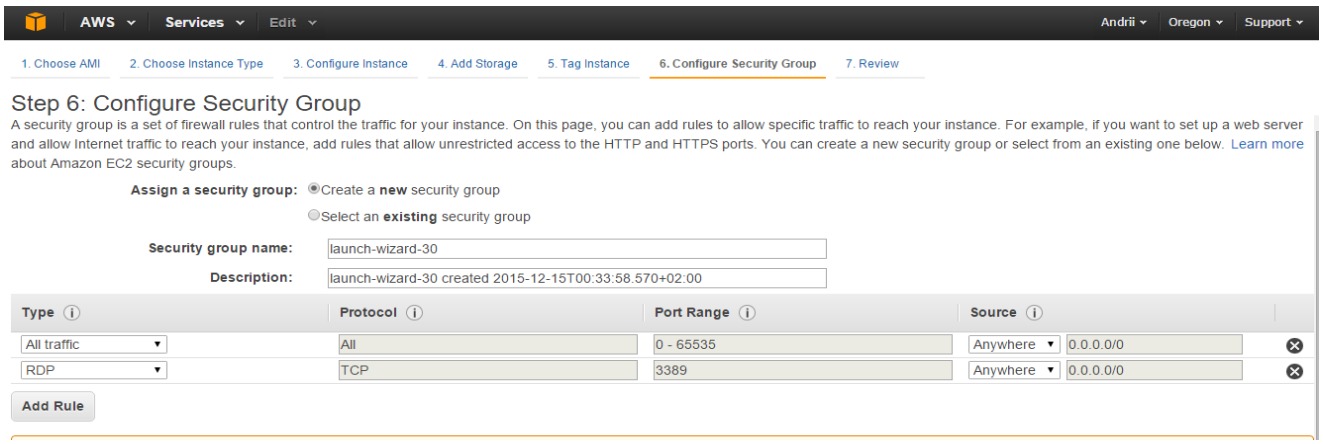

Рис.5. Налаштування мережі

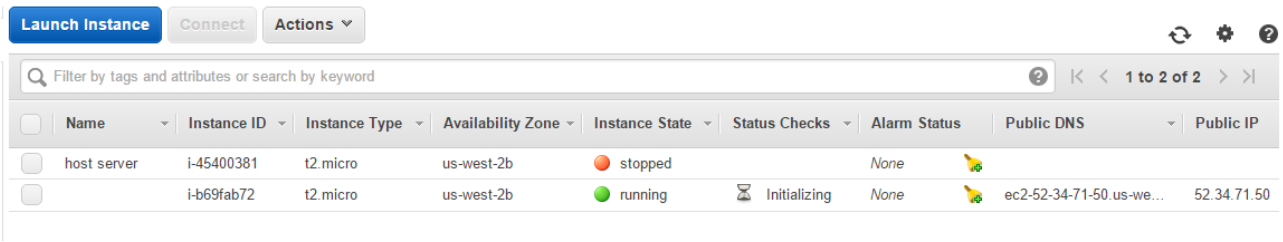

Рис.6. Працюючі VM

Для спрощення дозволимо будь-якому трафіку проходити через мережевий екран для будь-якого порту Рис.5.

Також ці правила потрібно продублювати в налаштуваннях безпеки Windows після створення віртуальної машини.

Після того, як всі налаштування обрано, потрібно натиснути на кнопку «Review and Launch», де можна перевірити правильність введених налаштувань. Після цього, потрібно натиснути на кнопку «Launch». На екрані з'явиться діалог, що дозволяє створити пару ключів для отримання і розшифрування паролю до машини.

Після завантаження ключа і натискання на кнопку «Launch Instances» починається створення віртуальної машини. Тепер, якщо повернутися до початкового меню, що показує список віртуальних машин, можна побачити нову машину в стані «running», тобто така, що запущена і працює Рис.6.

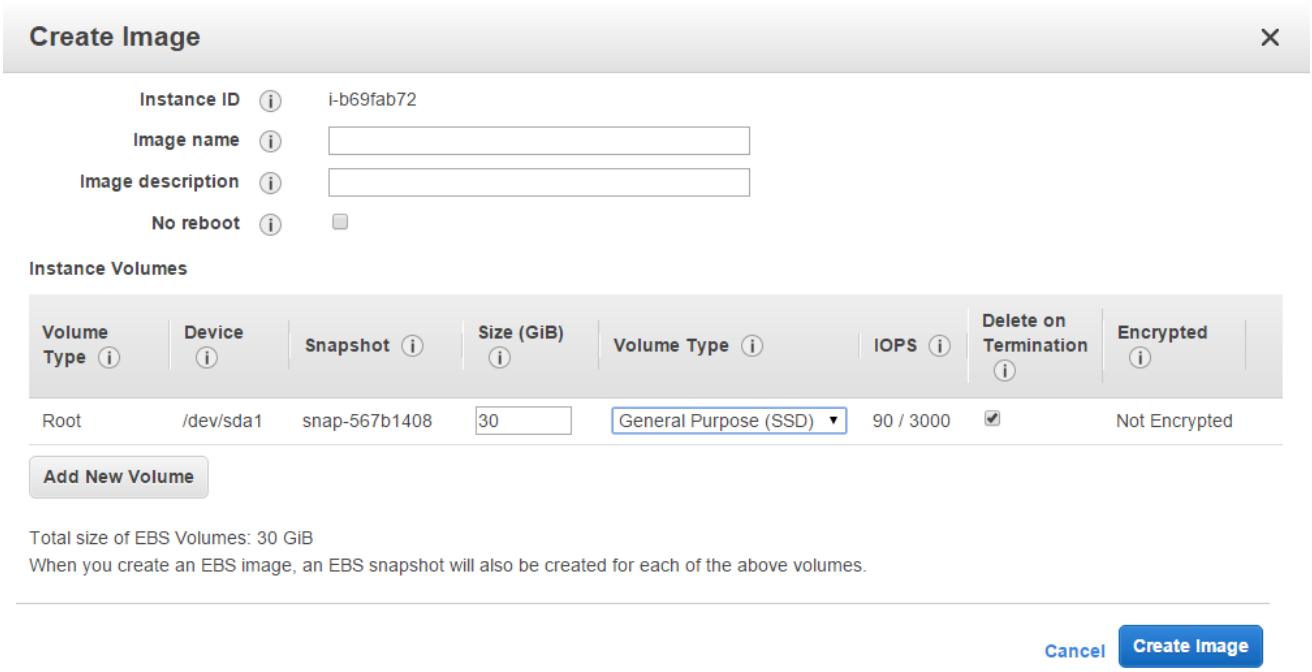

Рис.7. Створення образу машини

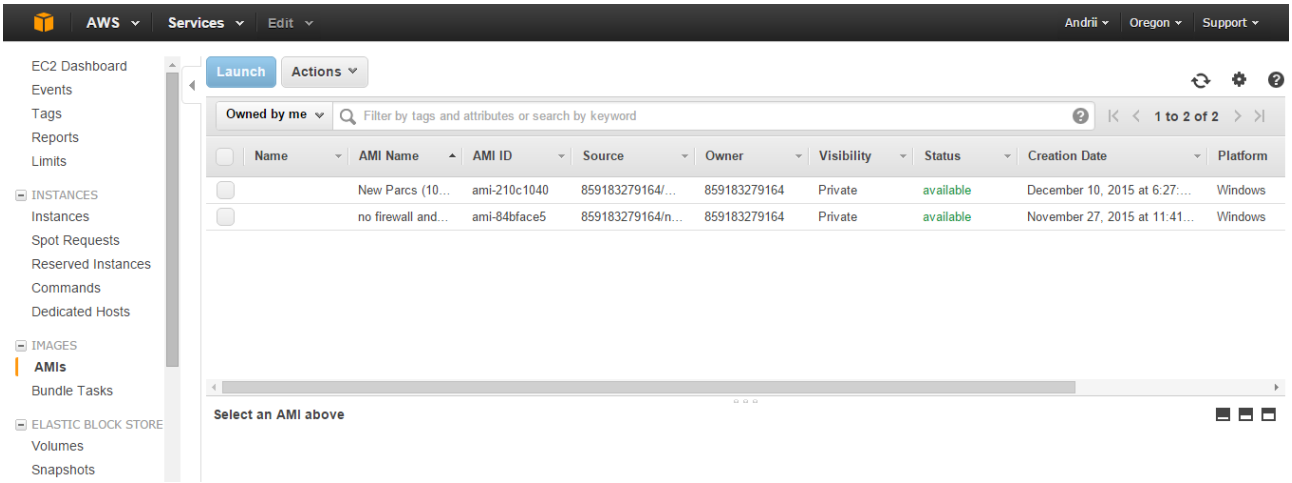

Рис.8. Відображення нового образу

Машині надається публічні DNS і IP-адреса. Для того, щоб працювати з новою віртуальною машиною, необхідно до неї підключитися. Для цього потрібно обрати машину і натиснути на кнопку «Connect». Після Натиснувши кнопку «Get Password», з'явиться діалог для отримання паролю до машини. Для цього потрібно використати завантажений на попередніх кроках ключ. Варто зазначити, що якщо машина була створена з власного образу, то пароль на неї буде такий же, як і на машину, з образу якої вона була створена. Отриманий пароль потрібно використати для підключення до машини через Remote Desktop Protocol (RDP).

Для цього потрібно завантажити файл підключення через Remote Desktop, натиснувши кнопку «Download Remote Desktop File». Відкривши цей файл, потрібно ввести отриманий пароль. Після цього за допомогою захищеного RDP- з'єднання почнеться сеанс роботи з віртуальною машиною, з якою можна працювати так само, як і зі звичайною машиною.

Також важливо вміти створювати образи машин. Для цього потрібно натиснути правою кнопкою на потрібну машину, обрати пункт «Image» і «Create Image». В діалогу, що з'явиться, можна ввести ім'я образу та інші деталі Рис.7.

Після цього і натиснення на кнопку «Create Image», почнеться створення образу. Коли образ створиться, він з'явиться в меню «AMI» під вкладкою «Images» Рис.8.

Створити нову машину на основі образу можна за допомогою кнопки «Launch» на вкладці «AMI», або, як описано вище, обравши створений образ у якості образу нової машини.

Таким чином, було описано основні засоби налаштування WEB-сервісу Amazon Elastic Compute Cloud (EC2).

Використання системи ПАРКС.NET на Amazon Elastic Compute Cloud

Для використання системи ПАРКС.NET на Amazon Elastic Compute Cloud необхідно виконати наступні кроки:

1. Створити віртуальну машину і скопіювати на неї файли ПАРКС.NET. Також є сенс додати в автоматичний запуск програму *Daemon* для того, щоб не запускати її окремо на кожній машини, яка буде брати участь в обчисленнях.
2. Створити образ цієї машини.
3. Створити достатню кількість (у безкоштовній версії максимальна кількість – 20 машин) віртуальних машин на основі образу (інакше потрібно копіювати необхідні для роботи системи файли на кожну машину).
4. Запустити *Daemon* на всіх машинах, які будуть брати участь в обчисленнях.
5. Запустити *HostServer* на машині, яка буде відповідати за диспетчеризацію завдань.
6. Прописати в файлі hosts.txt, що лежить в одній директорії з *HostServer,* IP-адреси машин, на яких запущено *Daemon*. Це можуть бути як приватні, так і публічні IP-адреси. У випадку, якщо алгоритмічний модуль (АМ), який буде запускатися на системі, розташований не на одній зі створених віртуальних машин, потрібно використати публічні IP-адресу. Якщо АМ знаходиться на одній з цих машин, то можливо є сенс використовувати приватні IP-адреси, оскільки вони не змінюються після кожного перезапуску машини.
7. У файлі server.txt, що знаходиться в одній директорії з АМ, прописати IP-адресу *HostServer.*

Після виконання цих кроків можна запускати алгоритмічні модулі, які будуть обчислюватися з використанням системи.

Тестування системи ПАРКС.NET на Amazon Elastic Compute Cloud

Для тестування роботи системи був реалізований класичний алгоритм множення матриць. Обчислення проводились на Amazon Elastic Compute Cloud з використанням до 16 віртуальних машин типу «*Burstable Performance Instances*» з мінімальною потужністю - *Intel Xeon* процесор з одним ядром і частотою до 3.3GHz. Особливістю даного типу машини є те, що Amazon EC2 надає таким машинам базовий рівень продуктивності, а також можливість «вибухати» в моменти великої завантаженості на період часу, пов'язаний з часом перебування незайнятою. Результати наведені в таблиці 1.

Результати показують, що зі збільшенням кількості машин досягнуто покращення в часі до 9 разів. Також результати множення на великих матрицях показують, що даний тип машин не дуже підходить для великих обчислень через свою неможливість працювати на повній потужності протягом довгого часу. Для цього існують інші типи машин, що не доступні в безкоштовній версії.

Таблиця 1

Час (с.) множення матриць для різної кількості машин (процесорів)

Матриці / Машини	1000x1000	2000x2000	4000x4000	10000x10000
1	15.1	118.6	1098.5	-
2	8.2	61.5	571	-
4	4.7	39.2	330.5	-
8	3.5	20.6	180.7	8119.9
16	-	14.1	132.7	2635.3

Висновки

В даній роботі була представлена система для паралельних розподілених обчислень на комп'ютерній мережі *ПАРКС.NET*, а також можливість її використання для хмарних обчислень. Було показано, як налаштовувати систему *ПАРКС.NET* для хмарних обчислень із застосуванням сервісу Amazon Elastic Compute Cloud та проведено тестування системи на класичній задачі множення матриць. При виконанні обчислень на *Amazon Elastic Compute Cloud* із застосуванням 16 віртуальних машин було досягнуто покращення в часі в 8-9 разів. При збільшенні розмірності обчислень маємо збільшення ефективності застосування системи *ПАРКС.NET*, що на практиці дає досить непоганий результат.

Результати роботи можуть бути використаними при створенні різноманітних хмарних обчислювальних комплексів. Поєднання даних технологій може застосовуватися для розв'язування широкого класу задач, що потребують паралельної розподіленої обробки. На цьому роботи над вдосконаленням систем не завершується.

Як варіант безкоштовного використання хмарних обчислень у тестовому варіанті (більш ніж 20 обчислювальних вузлів) можливо використати підключення до *Amazon EC2* багатьох користувачів та застосування системи *ПАРКС.NET* для керування розподіленим паралельними обчисленнями.

Список використаних джерел

1. Анисимов А.В. Особенности ПАРУС-технологии /Анисимов А.В., Кулябко П.П. // Кибернетика и системный анализ. – 1993. – №3. – С.128-137.
2. Анисимов А.В. Система ПАРУС-JAVA для параллельных вычислений на компьютерных сетях / Анисимов А.В., Деревянченко А.В. // Кибернетика и системный анализ. – 2005. – №1. – С.25-36
3. Деревянченко О.В. Моделювання паралельних програм за допомогою системи ПАРКС-JAVA // Наукові записки НаУКМА, Комп'ютерні науки. – 2005. – С.47-58.
4. Анісимов А.В. Паралельне програмування із застосуванням технології ПАРКС-JAVA / Анисимов А.В., Деревянченко О.В. // К.: ТОВ «Компанія ВАІТЕ». – 2013. – 78с.
5. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. 3-е изд. // СПб.: Питер, 2012. –928 с.
6. Деревянченко О.В. Система паралельних обчислень на комп'ютерній мережі ПАРКС-.NET / Деревянченко О.В., Хавро А.Ю. // Матеріали СКІТ'14. – 2014. – С.118-119.
7. Деревянченко О.В. Системи паралельних обчислень на комп'ютерній мережі на базі технології ПАРКС // Вісник Київського національного університету імені Тараса Шевченка Серія фізико-математичні науки. – 2014. – №2. – С. 124-127.
8. Amazon Elastic Compute Cloud веб-сервіс // Ел. ресурс, режим доступу: http://aws.amazon.com

References

1. ANISIMOV A.V., KULYABKO P.P. (1993) Features PARCS-technology, Cybernetics and Systems Analysis # 3, pp. 128-137.
2. ANISIMOV A.V., DEREVIANCHENKO A.V. (2005) The System PARCS-JAVA for Parallel Computations on Computer Networks, Cybernetics and Systems Analysis # 1, pp. 25-36.
3. DEREVIANCHENKO O.V. (2005) Simulation of parallel programs using the system PARCS-JAVA, Scientific notes Kyiv-Mohyla Academy, Computer Science, pp.47 -58.
4. ANISIMOV A.V., DEREVIANCHENKO O.V. (2013) Parallel programming applying the technology PARCS-JAVA, Company VAITE, 78p.
5. RICHTER J. (2010) CLR via C#, *Third Edition* , Microsoft Press, 896 p.
6. DEREVIANCHENKO O.V. KHAVRO A.U. (2014) The system of parallel computing in computer network PARCS-.NET, Materials ACIT '14, pp. 118-119.
7. DEREVIANCHENKO O.V. (2014) Systems of parallel computing in computer network based on PARCS-technology, Bulletin of Taras Shevchenko National University of Kyiv, Series Physics & Mathematics # 2, pp. 124-127.
8. Amazon Elastic Compute Cloud: http://aws.amazon.com

Надійшла до редколегії 21.12.15