УДК 004.75

БАБЕНКО О. А.,
ГІОРГІЗОВА-ГАЙ В.Ш.

БАЛАНСУВАННЯ НАВАНТАЖЕННЯ ДОДАТКІВ У ПРИВАТНОМУ «ХМАРНОМУ» СЕРЕДОВИЩІ

У даній статті проведено порівняльну характеристику розповсюджених систем балансування навантаження Web додатків у відповідності до критеріїв, які враховують особливості роботи самих додатків і особливості інфраструктур обчислювальних комплексів, на яких вони розгортаються. Надано практичні рекомендації з налаштування системи балансування навантаження у «хмарній» платформі CloudStack на прикладі розгортання додатку дистанційного відео навчання. Приведено результати дослідження ефективності створених правил балансування.

This article shows comparative characteristics of the widespread load balancing systems for Web applications in accordance with criteria that take into account the characteristics of applications and computer systems infrastructure on which they are deployed. Practical recommendations for configuring the CloudStack platform load balancing system are given for example of remote video teaching application. The effectiveness of the created balancing rules investigated.

Ключові слова: балансування навантаження, Web додатки, CloudStack, хмарні технології

Вступ

Сьогодні все більше установ і організацій прагнуть створити власні Web сервіси як у мережі Інтернет так для корпоративних потреб на базі приватних або публічних «хмарних» інфраструктур. Оскільки такі «хмарні» системи належать до високонавантажених, то можливості масштабування і балансування навантаження при їх побудові мають особливе значення. Від вибору і налаштування системи балансування навантаження, які б найкращим чином відповідали задачам створення конкретного сервісу в конкретних умовах, напряму залежить ефективність роботи сервісу і безперебійність надання послуг.

Мета статті

Метою даної статті є надання практичних рекомендацій з вибору систем балансування навантаження Web додатків на основі критеріїв, які враховують особливості роботи самих додатків і особливості інфраструктур обчислювальних комплексів, на яких вони розгортаються, а також – з налаштування системи балансування навантаження у відкритій «хмарній» платформі CloudStack на прикладі організації сервісу дистанційного відео навчання. Для цього у статті проведено порівняльну характеристику ряду розповсюджених систем балансування навантаження Web додатків у відповідності до обраних критеріїв, та результати дослідження ефективності правил балансування, створених для «хмарного» сервісу.

Порівняльна характеристика розповсюджених систем балансування навантаження

Балансування (вирівнювання) навантаження, (англ. Load balancing) – метод розподілу завдань між декількома мережевими пристроями (наприклад, серверами) з метою оптимізації використання ресурсів, скорочення часу обслуговування запитів, горизонтального масштабування кластера (динамічне додавання / видалення пристроїв), а також забезпечення відмовостійкості (резервування).

При балансуванні навантаження «хмарних» додатків слід враховувати особливості роботи додатків, особливості інфраструктури обчислювальних комплексів і додаткові вимоги при організації сервісів.

Для порівняння можливостей систем балансування навантаження додатків в "хмарних" середовищах визначимо основні критерії такого порівняння.

Ефективність розподілу навантаження – характеризує можливості системи забезпечити рівномірність розподілу навантаження між обслуговуючими вузлами при різних вимогах до роботи додатків і до структур обчислювальних комплексів, які можуть бути враховані через наявні в сис-

темі алгоритми балансування навантаження і тип балансування (статичне і динамічне).

Основною перевагою статичного розподілу, який не змінюється у процесі роботи сервісу, є простота реалізації. Динамічне балансування є більш гнучким і передбачає перерозподіл навантаження на обчислювальні вузли під час роботи сервісу на підставі визначення прямих показників (завантаження центрального процесора, обсяг доступної оперативної пам'яті, доступного дискового простору тощо), або непрямих (кількість активних підключень до сервера чи час його відгуку).

Врахування гео-положення вузлів – характеризує можливість системи забезпечити зниження часу її відгуку через врахування географічної близькості певних груп клієнтів до певних груп обслуговуючих вузлів. Потрібно зауважити, що ефективність методу дуже залежить від актуальності даних про довжину маршрутів, пропускну спроможність і завантаження ліній зв'язку між потенційними клієнтами і серверами.

Незалежність від типу мережі – незалежність системи балансування від того, у локальній чи у глобальній мережі знаходяться обслуговуючі сервери, а також незалежність від типу серверної платформи.

Можливості автоматичного масштабування – характеризує наявність в системі механізмів горизонтального масштабування обслуговуючих вузлів (динамічне додавання/видалення) при балансуванні навантаження. Автомасштабування в «хмарних» платформах забезпечується спеціальними модулями, зв'язаними з системами балансування навантаження, які безпосередньо цю задачу не вирішують.

Автоматичне визначення і видалення непрацюючих вузлів – забезпечення системою балансування відмово стійкості сервісу на основі моніторингу доступності і працездатності його компонентів та автоматичного видалення непрацюючих вузлів з пулу розподілу навантаження.

Забезпечення довгострокових сесій – характеризує можливість системи балансування забезпечити додатку зберігання стану сесії користувача з декількох повторних запитів. Може реалізуватися в системі або через з'єднання користувача з одним і тим самим сервером, або через зберігання сесії в загальному для всіх серверів сховищі. Прив'язка клієнта до конкретного сервера зазвичай реалізується або за його IP-адресою (яка, однак, може змінюватись через певний час), або по cookies (в який записується ідентифікатор сервера).

Незалежність від прикладного протоколу – характеризує незалежність методу балансування навантаження від використовуваного протоколу прикладного рівня до будь-яких TCP- сервісів: HTTP, HTTPS, SMTP, IMAP, SSH, FTP, SQL і т.д. Це досягається за рахунок балансування запитів на 2-4 рівнях моделі OSI. Найчастіше використовується балансування на 4-му рівні OSI (IP- адреси + номери портів TCP і UDP), як найбільш інформативне і гнучке.

Контентно залежний аналіз, кешування – характеризує можливості системи забезпечити балансування за видом контенту по різних серверах (наприклад, до статичних сторінок, до динамічних сторінок, до медіа серверів і т.д.). Оскільки така можливість з'являється тільки при балансуванні на 7-му рівні моделі OSI за допомогою кешуючих проксі серверів, то можливості системи балансування розширюються за рахунок кешування відповідей, часто самостійної обробки HTTPS трафіку, стискання даних, що знижує навантаження обслуговуючих серверів, аналізу запитів для підвищення безпеки сервісу і т.д. До недоліків методу є високе споживання ресурсів і залежність від прикладного протоколу (в даному випадку розглядається тільки протокол HTTP).

Таким чином, при виборі системи балансування навантаження потрібно чітко розуміти задачі, які вона повинна вирішувати для конкретного сервісу, і наявність в ній засобів, які будуть для цього ефективними.

У в таблиці 1 наведено порівняльну характеристику ряду розповсюджених систем балансування навантаження.

Табл. 1. Порівняльна характеристика розповсюджених систем балансування навантаження

	Google Compute Engine	**Microsoft Azure Load Balancer / LoadMaster**	**Amazon EC2 Elastic Load Balancing**	**HAProxy**	**LVS**	**Nginx**	**Cloud-Stack**
Ефективність розподілу навантаження	+	-/+	+	+	+	+	+
Врахування гео-положення вузлів	+	-/+	+	-	+	-	+

	Google Compute Engine	Microsoft Azure Load Balancer / LoadMaster	Amazon EC2 Elastic Load Balancing	HAProxy	LVS	Nginx	Cloud-Stack
Можливості автомасштабування	+	+	+	-	-	-	+
Незалежність від прикладного протоколу (4-й рівень OSI)	+	+	+	+	+	-	+
Контентно залежне балансування, аналіз, кешування (7-й рівень OSI)	+	-/+	+	+	-	+	-
Забезпечення довгострокових сесій	+	-/+	+	+	+	+	+
Незалежність від типу мережі	+	+	+	+	+	-	+
Автоматичне визначення і видалення непрацюючих вузлів	+	+	+	+	-+	+	+

Система балансування Elastic Load Balancing, яка використовується в Amazon EC2 [1] є найбільш функціональною з розглянутих систем, в ній є все те, що зазвичай вимагається від системи балансування: автомасштабування, відмовостійкість, контроль за доступністю обслуговуючих серверів, простота використання, гнучкість, надійність, безпека, можливість міграції завдань, врахування явних параметрів завантаження обслуговуючих серверів. За рахунок великого рівня автоматизації, людське втручання в роботу системи зведено до мінімуму.

Система балансування Google Compute Engine [2] також має достатньо повний набір функціональних можливостей. Хоча система є менш гнучкою і універсальною, а також гірше документованою, у порівнянні з системою Amazon EC2, проте, її перевагами є більша простота використання і більш висока швидкодія програмно-апаратного рішення.

На цьому фоні дивує бідність можливостей вбудованої системи балансування Azure Load Balancer [3], хоча більшість її обмежень сьогодні дозволяє подолати рішення від Kemp Technologies LoadMaster-for-Azure [4]. Система LoadMaster забезпечує повноцінне балансування навантаження на 4-7 рівні моделі OSI для робочих навантажень в «хмарному» середовищі Azure, перевірку працездатності додатків і прискорення SSL, а також додаткові послуги, такі як запобігання вторгнень, кешування і стиснення даних для опублікованих служб. LoadMaster оптимізує потік трафіку для множинних віртуальних служб, багатокомпонентних додатків, розробка і розгортання яких в Microsoft Azure є перевагами цієї «хмарної» платформи так само як і висока продуктивність масштабних обчислень. Об'єкти LoadMaster, розміщені в хмарі Microsoft Azure і в середовищах локальної приватної хмари, працюють спільно для забезпечення безперервності доставки послуг на міжхмарному рівні.

Система HAProxy це безкоштовне, дуже швидке, ефективне (в плані використання процесора і оперативної пам'яті) і надійне рішення, що пропонує високу доступність і балансування навантаження для розподілених TCP і HTTP-додатків. Система HAProxy також є складовою частиною «хмарної» платформи Red Hat OpenShift і балансувальником по замовчанню у частиною «хмарній» платформі OpenStack [5].

Система Linux Virtual Server (LVS) - це безкоштовний популярний засіб управління кластерними системами для Linux. LVS дозволяє створити кластер з фізичних (або віртуальних) серверів, що розподіляє навантаження між машинами в залежності від їх стану, пріоритету та інших параметрів настроювання завдяки великій кількості алгоритмів балансування [6]. Система підтримує балансування навантаження на 4-му рівні моделі OSI, балансування на 7-му рівні на сьогодні не допрацьовано. До недоліків LVS можна віднести складність налаштування, необхідність встановлення додаткових пакетів для забезпечення високої готовності сервісу, відмінності в налаштуваннях для різних версій Linux. Найбільш докладно описано налаштування для RedHat Enterprise Linux Редакция 6, наприклад, у [7].

Система Nginx – це популярне безкоштовне ПЗ, що реалізує HTTP-сервер і зворотний проксі сервер, поштовий проксі-сервер, а також TCP проксі-сервер загального призначення [8]. Nginx забезпечує всі переваги, які надають

HTTP проксі сервери, і використовувати його доцільно при необхідності детального настроювання всіх, навіть найдрібніших аспектів балансування HTTP і HTTPS трафіку. Системам HAProxy та LVS Nginx програє по швидкодії та використанню ресурсів на балансування.

Вбудована система балансування відкритої «хмарної» платформи CloudStack 4 забезпечує досить повний набір функціонально можливостей, хоча і реалізує балансування навантаження тільки на 4-му рівні. Це обмеження, при необхідності, легко можна подолати, якщо прийняти до уваги, що платформа CloudStack підтримує зовнішні система балансування навантаження. В їх якості можна використовувати як комерційні балансувальники від Google, Microsoft, Amazon, Rackspace, NetScaler, так і відкриті, такі як HAProxy, LVS. Сьогодні CloudStack – досить потужна та надійна платформа з простим інтуїтивно зрозумілим інтерфейсом, яка багато чого запозичила у Amazon EC2, і дозволяє організувати роботу як публічного IaaS-сервісу, так і приватної «хмарної» інфраструктури [9].

Створення правил балансування в системі CloudStack

В якості прикладу розглянемо організацію сервісу дистанційного відео навчання для студентів заочної форми навчання і другої післядипломної освіти, активність яких сильно змінюється протягом періоду навчання. Сервіс розгортається на локальних серверах у приватному «хмарному» середовищі кафедри СП ІПСА КПІ на базі відкритого програмного забезпечення BigBlueButton [10].

Система BigBlueButton підтримує наявність декількох аудіодоріжок і обмін відео, можливість показу презентацій, документів Microsoft Office і OpenOffice, зображень, PDF документів, підтримуються розширені можливості дошки. Для зворотного зв'язку зі слухачами веб-конференції існують публічні та приватні чати. Сервер BigBlueButton може працювати в хмарному середовищі, як Amazon EC2, при його встановленні під ОС Ubuntu 14.04 64-бітної версії.

Встановлення сервісу складається з наступних кроків, детальний опис яких виходить за рамки даної статті:

- Створення шаблону з Ubuntu 14.04 64 біт;
- Створення віртуальної машини з шаблону і налаштування її;
- Встановлення сервера BigBlueButton на віртуальну машину;
- Створення шаблону з готовою віртуальну машину;
- Створення другої копії віртуальної машини.

На основі вимог додатку і у відповідності до умов розгортання сервісу сформулюємо правила для системи балансування, яка буде розподіляти трафік, отриманий на відкритій IP- адресі сервісу, на дві віртуальні машини. При створенні правила необхідно визначити алгоритм балансування, політику липкості, задати необхідні параметри і присвоїти правило множині віртуальних машин. Також треба визначити політику перевірки доступності серверів і авто масштабування.

Вибір алгоритму. Серед Round Robin (рівномірний статичний розподіл), Source Ip Hash (розподіл клієнтів по серверах на основі близькості) і Least Connection (динамічний розподіл на сервери з найменшою кількістю з'єднань) вибираємо алгоритм Least Connection, так як всі сервери знаходяться у одній локальній мережі, а при роботі з додатком передбачається зберігання стану сесій користувачів, що може призвести до нерівномірного навантаження при використанні статичних алгоритмів.

Політика липкості. Ця політика потрібна для ідентифікації користувача при повторних з'єднаннях і пошуку збережених ним даних на тому ж самому сервері. З двох методів – за cookie (згенерованим системою балансування або додатком) або IP-адресою джерела, вибираємо за IP-адресою користувача. Оскільки не передбачається сесія довше одного робочого дня, то у переважній більшості випадків динамічна IP-адреса користувача за цей час не зміниться.

Оскільки у період активного використання сервісу потрібно забезпечити його високу готовність, задаємо перевірку «стану здоров'я» серверів з опитуванням їх доступності кожні 10 секунд.

Налаштуємо політику авто масштабування, оскільки передбачається, що активність користувачів буди сильно змінюватись у часі. Масштабування вгору будемо здійснювати, коли використання CPU або RAM серверів почне перевищувати 80%, а вниз – коли використання CPU або RAM буде менше ніж 20%.

Для додавання правила балансування навантаження виконаємо наступні кроки:

1. У лівій панелі навігації графічного інтерфейсу CloudStack вибираємо вкладку "Мережа".

2. Натискаємо на назву нашої мережі, де ми хочемо балансувати трафік. Назва - default.

3. Натискаємо кнопку "Перегляд IP-адрес".

4. Натискаємо IP-адресу, для якої хочемо створити правило (192.168.1.125), і вибираємо вкладку "Конфігурація".

5. У вузлі балансування навантаження, натискаємо "Показати всі".

У базовій зоні, також можна створити правило балансування навантаження без вибору IP-адреси. CloudStack сам призначить IP- адреси серверам при створенні правила балансування навантаження, які будуть вказані на сторінці IP-адреси створення правила. Щоб зробити це, треба вибрати ім'я мережі, а потім натиснути вкладку "Додати балансувальник навантаження".

6. У вікні, що відкриється (рис. 1), потрібно заповнити наступні поля:

- Ім'я (Name): web;
- Публічний порт (Public Port): 80;
- Приватний Порт (Private Port): 80;
- Алгоритм (Algorithm): Least connection;
- Липкість (Stickiness): Метод SourceBased.

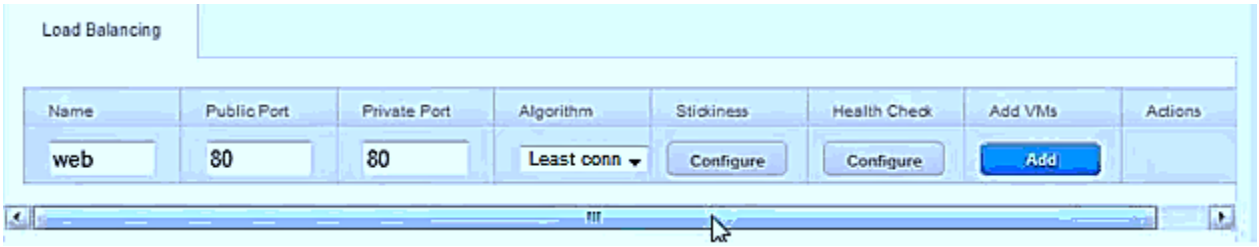

Рис. 1. Налаштування правила балансування CloudStack

7. На вкладниці Stickiness, потрібно задати метод політики липкості (load balancer-generated cookie, application-generated cookie, чи source-based), ім'я політики та додаткові параметри в залежності від методу (розмір таблиці IP адрес (200 тис. записів), строк їх дії (50 годин)) (рис. 2).

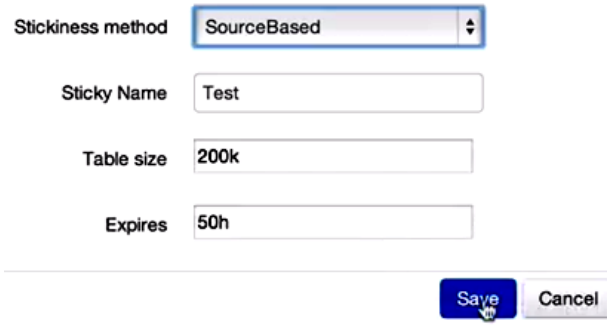

Рис. 2. Вікно конфігурації Stickiness CloudStack

8. На вкладниці Health Check (Перевірка здоров'я), натискаємо кнопку "Налаштування" (рис. 3) і заповняємо визначені нами параметри перевірки доступності серверів:

- Ping path: Послідовність адрес, до яких слід відправити запити перевірки. Залишаємо значення за замовчуванням: / (all).
- Response time: Час очікування відповіді від адрес. Задамо 10 сек.
- Interval time: Час між перевірками здоров'я. Задамо 10 секунд.
- Healthy threshold: Число послідовних успіхів перевірок здоров'я, які необхідні перш ніж оголосити екземпляр здоровим. Задамо 1.
- Unhealthy threshold: Число послідовних невдач перевірки здоров'я, які необхідні, перш ніж оголосити екземпляр нездоровим. Задамо 1.

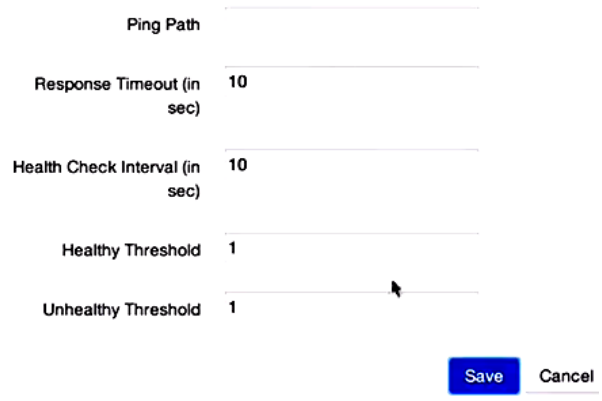

Рис. 3. Вікно конфігурації перевірки здоров'я CloudStack

9. Створюємо правила автомасштабування.

Налаштуємо масштабування вгору при використанні CPU або RAM більше 80% та масштабування вниз при використанні CPU або RAM менше ніж на 20% (рис. 4). Заповнюємо наступні поля:

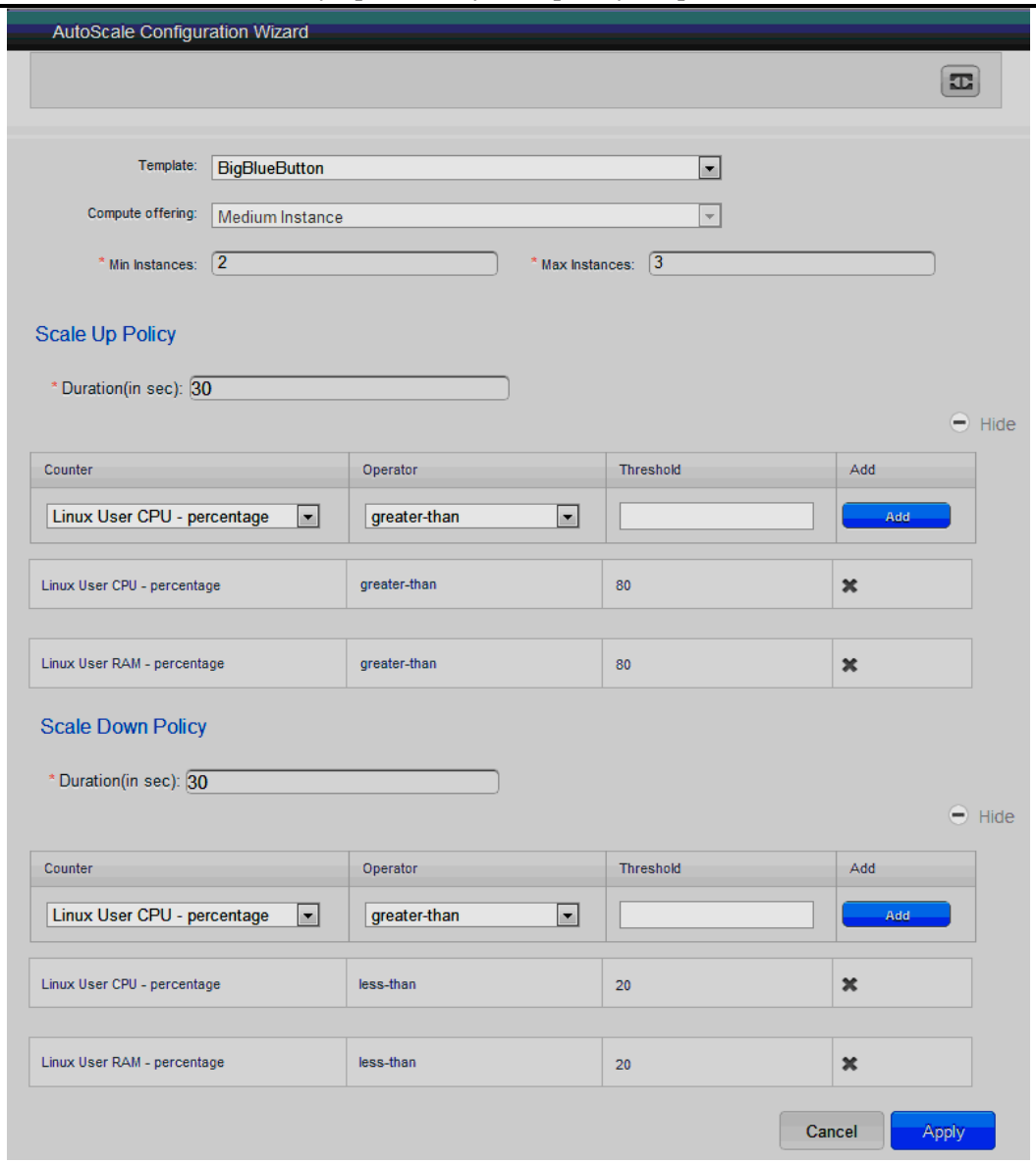

Рис. 4. Правила автомасштабування CloudStack

10. Натискаємо кнопку "Додати віртуальні машини", а потім вибираємо наші дві віртуальні машини зі встановленим BigBlueButton, які будуть розділяти навантаження вхідного трафіку, і натискаємо кнопку "Застосувати".

У вікні на рис. 5 бачимо, що нове правило балансування навантаження з'явилося в списку.

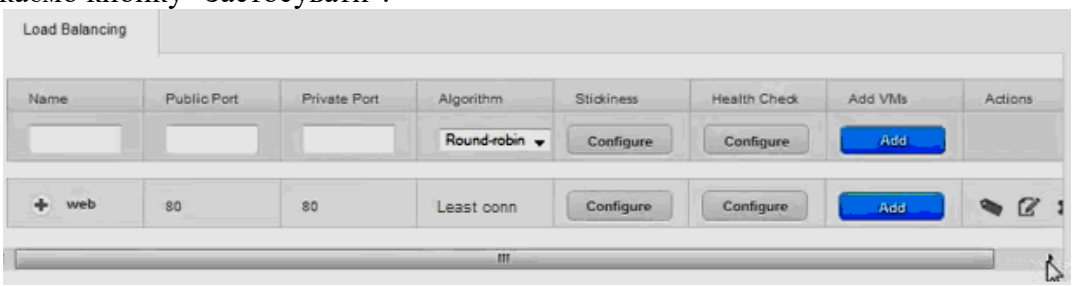

Рис. 5. Список створених правил балансування CloudStack

Навантажувальне тестування

Основна мета навантажувального тестування полягає в тому, щоб, створивши певне очікуване в системі навантаження (наприклад, за допомогою віртуальних користувачів) і використавши ідентичне програмне і апаратне забезпе-

чення, спостерігати за показниками продуктивності системи.

В якості засобу для навантажувального тестування було обрано WAPT (Web Application Performance Testing), який є засобом навантажувального і стресового тестування і надає можливості для детального та ефективного тестування веб-сайтів і Інтернет додатків з веб-інтерфейсом [11].

Продукт WAPT може емулювати типову активність тисяч користувачів, що працюють з сайтом одночасно. Він наділений низкою спеціальних можливостей, які дозволяють тестувати RIA-додатки з динамічними даними, що змінюються прямо під час тесту. WAPT підтримує всі типи авторизації на сервері. Результати тестування представляються у наочній формі у вигляді звітів і графіків, які дозволять проаналізувати характеристики продуктивності сайту під навантаженням різних типів і рівнів, виявити і усунути вузькі місця у роботі сайту, оптимізувати конфігурацію обладнання та програмного забезпечення.

Для налаштування тестування в WAPT нам необхідно:

- створити сценарій тесту за допомогою внутрішнього браузера програми, який записує дії користувача у веб-додатку;
- налаштувати кількість користувачів та час роботи тестування;
- додати два наших сервера в засіб моніторингу;
- додати агент для тестування.

При кожному тестуванні будемо імітувати роботу користувача (логін, переміщення в інтерфейсі системи) з використанням веб-камери і мікрофону.

В першому тесті перевіримо ефективність балансування. Для нього задамо в налаштуваннях тестування кількість користувачів, що збільшується від 0 до 40 з додаванням по 2 користувача кожні 10 секунд. Результати можна побачити на рис. 6 (для першої віртуальної машини) та на рис. 7 (для другої віртуальної машини).

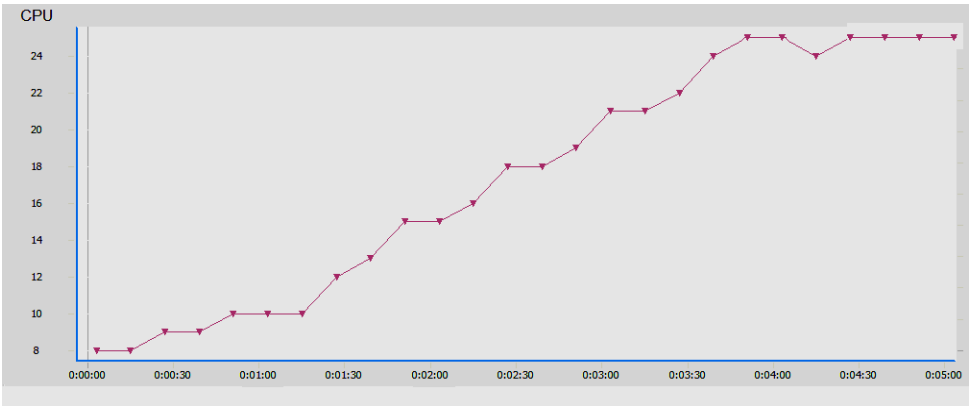

Рис. 6. Графік навантаження на CPU для першої віртуальної машини

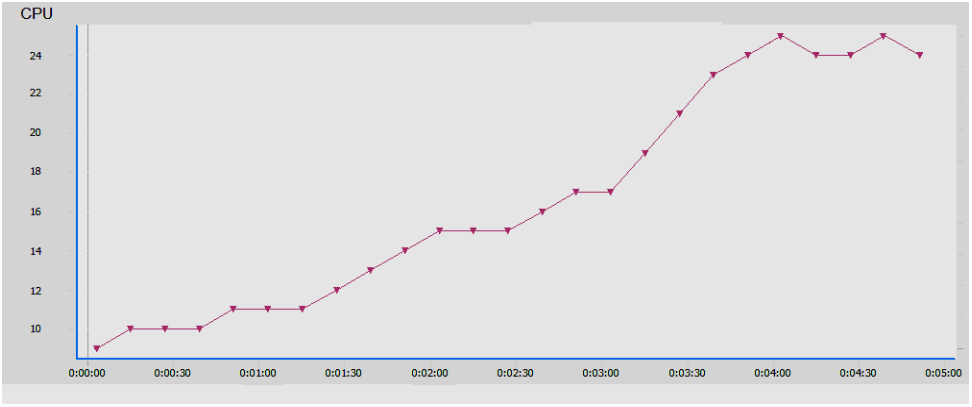

Рис. 7. Графік навантаження на CPU для другої віртуальної машини

З результатів тестування видно, що система балансування справляється з навантаженням і досить рівномірно розподіляє навантаження на обидві віртуальної машини. В момент часу 0:30:20 надходять останні користувачі, після чого навантаження вирівнюється. Продивившись логи CloudStack, бачимо, що на кожну машину прийшло по 20 користувачів.

Далі проведемо тестування роботи засобу автомасштабування CloudStack. Для цього зробимо два сценарії тесту, які будуть виконуватись послідовно. В першому сценарії кількість користувачів буде збільшуватися від 0 до 160 з додаванням 2 користувачів кожні 6 секунд. В другому – ті ж користувачі, створені в першому сценарії будуть завершувати сесію і їх кількість буде зменшуватись від 160 до 0 по 2 користувача кожні 6 секунд. Результати наведені на рис. 8 (для першого сценарію) і на рис. 9 (для другого сценарію).

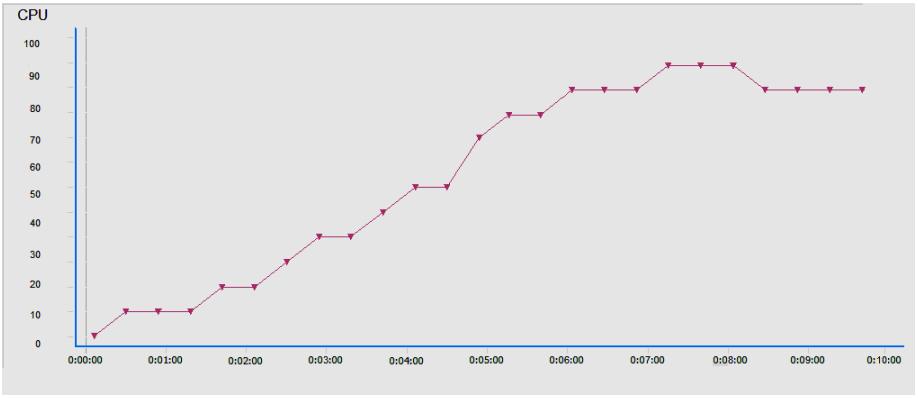

Рис. 8. Графік навантаження на CPU першої віртуальної машини при виконанні першої половини сценарію

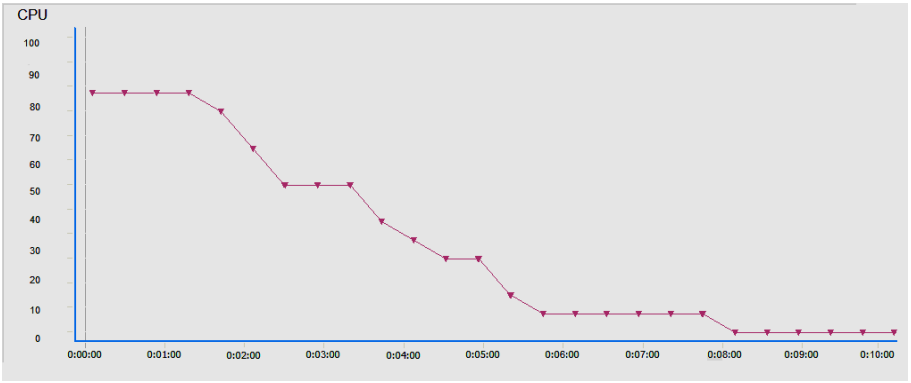

Рис. 9. Графік навантаження на CPU першої віртуальної машини при виконанні другої половини сценарію

Продивившись логи CloudStack, бачимо, що на наші дві машини прийшло по 69 користувачів. В момент часу 00:05:43 навантаження на CPU перевищило 80%, після чого через 30 секунд в CloudStack було ініційоване створення нової віртуальної машини, яка почала працювати в момент часу 00:06:35. Новий список віртуальних машин можна побачити на рис. 10.

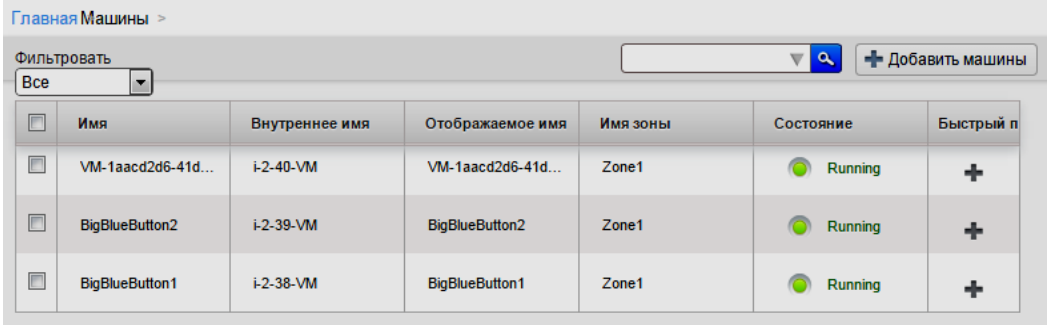

Рис. 10. Список віртуальної машини після активації засобу автомасштабування в CloudStack

Всіх наступних користувачів, відповідно до алгоритму Least Connections, балансувальник відправляв на нову машину. Кількість користувачів на ній досягла 22. Як можна побачити з

графіка (рис. 8) після створення нової машини, навантаження на першій машині вирівнюється.

Наступна фаза тестування почалась після моменту часу 00:10:00 (з онуленням таймера). Під час тесту навантаження на першій машині спадає, поки всі сесії користувачів не будуть завершені, а потім вирівнюється. В момент часу 00:05:16 навантаження на CPU стало нижче 20%, після чого через 30 секунд було ініційоване знищення третьої віртуальної машини (коли всі сесії її користувачів були завершені). Список віртуальних машин після завершення тестування знову складається з двох машин, які були створені вручну (рис. 11).

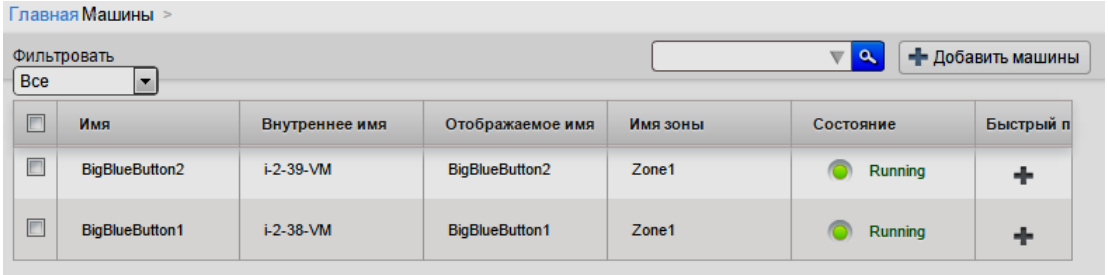

Рис. 11. Список ВМ CloudStack після завершення тестування

Висновки

Таким чином, результати навантажувального тестування розгорнутого «хмарного» сервісу показали ефективність створених правил балансування та роботи засобу автомасштабування. Система CloudStack показала себе як досить надійна та потужна платформа для створення приватної "хмари" корпоративного рівня та балансування навантаження всередині неї.

Результати проведених досліджень можуть бути корисні для вибору і налаштування систем балансування навантаження при створенні "хмарних" сервісів корпоративного рівня.

Список посилань

1. AWS | Elastic Load Balancing– сетевой балансировщик нагрузки в облаке. – Режим доступу: http://aws.amazon.com/ru/elasticloadbalancing/ – Дата доступу: 04.05.2015
2. Load Balancing – Compute Engine – Google Cloud Platform. – Режим доступу: https://cloud.google.com/compute/docs/load-balancing/ – Дата доступу : 04.05.2015
3. Azure Load Balancer overview | Microsoft Azure. – Режим доступу: https://azure.microsoft.com/ru-ru/documentation/articles/load-balancer-overview/ – Дата доступу: 04.05.2015
4. LoadMaster For Azure. – Режим доступу: http://kemptechnologies.com/solutions/microsoft-load-balancing/loadmaster-azure/ – Дата доступу: 04.05.2015
5. Load Balance OpenStack API – RDO. – Режим доступу: https://www.rdoproject.org/Load_Balance_OpenStack_API – Дата доступу: 04.05.2015
6. LVS Introduction – Load Balancing Server Cluster. – Режим доступу: http://www.linuxvirtualserver.org/whatis.html – Дата доступу: 05.05.2015
7. Комплект распределения нагрузки для Red Hat Enterprise Linux Редакция 6 – Режим доступу: http://adm-lib.ru/books/sergey/RHEL/docs/Virtual_Server-6.pdf - Дата доступу: 25.06.2015
8. nginx. – Режим доступу: http://nginx.org/ru/ – Дата доступу: 05.05.2015
9. Apache CloudStack 4 – News.stfw.ru. – Режим доступу: http://news.stfw.ru/716-apache-cloudstack-4.html – Дата доступу: 05.05.2015
10. BigBlueButton. – Режим доступу: http://docs.bigbluebutton.org – Дата доступу: 02.05.2015
11. WAPT – Инструмент для Нагрузочного Тестирования Сайтов и Веб-приложений. – Режим доступу: http://www.loadtesting.ru/product.shtml – Дата доступу: 26.05.2015