БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В ХМАРНИХ ОБЧИСЛЕННЯХ ЦОД CALL CENTER

Лук'янов О., Наритник Т.М., Сабурова С.О.

Кафедра інфокомунікаційної інженерії ім.В.В.Поповського,
Харківський національний університет радіоелетроніки,
Україна
Кафедра телекомунікацій,
Національний технічний університет «КПІ імені Ігоря Сікорського»,
Україна
E-mail: oleh.lukianov@nure.ua,
t.narytnyk@ukr.net

Abstract

The paper presents cloud technologies that are based on the use of a distributed system to provide services to subscribers from a personal computer to distributed virtual computer clusters: virtual machines and hosts with access via the high-speed Internet. The possibilities of efficient use of cloud computing resources in the Call Center data processing center in the conditions of ensuring elasticity and scalability at the level of service level requirements (SLA) and quality system (QoS) are investigated. Algorithms are presented for analyzing the state of hosts and virtual machines under traffic overload and load conditions based on different resource parameters for performing VM migration tasks while balancing the overall load. Due to the efficient use of resources, the algorithm not only saves energy but also improves the quality of calculations. These parameters are also used to check whether the proposed load balancing is normalized enough to balance the Call Center data center load.

На цей час хмарні обчислення стають дуже популярними через їх широке використання для надання ресурсів зовнішнім користувачам. У результаті хмара стала найперспективнішою технологією як у промисловості, так і в наукових колах. Ця модель, заснована на використанні розподіленої системи для розподілу завдань користувача з резидентного комп'ютера на віддалені віртуалізовані комп'ютерні кластери, ізольовані від продуктивності, відомі як віртуальні машини (ВМ). Щоб отримати віртуальні машини від хоста та надати його користувачам, потрібен високошвидкісний Інтернет.

Хмара надає свої послуги дуже гнучким способом, що забезпечує еластичність і масштабованість ресурсів, дотримуючись моделі оплати за використання. Комерційні хмарні постачальники, такі як Google, Amazon, Yahoo та Microsoft, надають такі послуги своїм користувачам по всьому світу. Методи віртуалізації ефективно використовуються для спільного використання фізичної машини (хоста) між декількома ізольованими за продуктивністю платформами. Постачальники хмарних послуг (Cloud Service Providers, CSP) відповідають за надання хмарних послуг своїм користувачам. По суті, постачальники послуг відповідають за своєчасне надання процесора, пам'яті та програмного забезпечення як послуг своїм кінцевим користувачам.

Хмара також надає послуги центру обробки даних (ЦОД) Call Center з підтримкою лічильників, де не лише кінцеві користувачі управляють зменшенням капітальних витрат на апаратне чи програмне забезпечення, зберігаючи при цьому вимоги системи QoS. Таким чином, хмарні обчислення задовольняють наступні потреби ЦОД Call Center.

1) Динамічность: одна з найважливіших потреб обчислювальних ресурсів, які можуть бути збільшені або зменшені через неоднорідний попит користувачів.

2) Абстракція: кінцеві користувачі повністю абстраговані від основних функцій хмарних обчислень, їм не потрібно піклуватися про операційні системи (ОС), плагіни, веб-безпеку, програмне забезпечення чи платформу. Всі ці речі повністю абстраговані від кінцевих користувачів.

3) Спільне використання ресурсів: хмара ділиться своїми ресурсами, не лише забезпечуючи оптимальне використання ресурсів, але й роблячи ресурс адаптивним для спільного використання додатків і багатогранних мережних ресурсів.

Вищезазначені потреби повинні бути забезпечені CSP у межах, передбачених для забезпечення ефективного хмарного обслуговування своїх користувачів за вимогами QoS. Крім того, угода про рівень обслуговування (Service Level Agreement, SLA) підписуються між користувачами CSP і хмари. Існує багато проектів, які розроблені та перевірені для надання SLA, у т.ч. фунціонування Call Center нових поколінь.

В свою чергу балансування навантаження – це процес призначення навантаження на окремий вузол із загальної системи центру обробки даних (ЦОД) Call Center, щоб зробити використання ресурсів більш ефективним і покращити час відповіді на виклик та завдання користувача. Алгоритми балансування навантаження одночасно забезпечують рівномірний розподіл робочих навантажень, усуваючи умови перевантаження або недовантаження на всіх вузлах, які можуть виникнути в хмарному середовищі. Через динамічний характер не можливо передбачити кількість запитів, які надсилаються що секунди.

Ця непередбачувана природа є постійно мінливою поведінкою хмари. Якисний алгоритм балансування навантаження дуже потрібен, щоб знайти спосіб максимізувати використання ресурсів і підвищити пропускну здатність, продуктивність, масштабованість, відмовостійкість і час відгуку ЦОД Call Center. Завдяки ефективному використанню ресурсів алгоритм не тільки економить енергію, але й підвищує якість обчислень. Ці параметри також використовуються для перевірки того, чи запропоноване балансування навантаження достатньо нормовано, щоб збалансувати навантаження ЦОД Call Center.

Особливості хмарних обчислень:

- навантаження на ключові ресурси носить періодичний і нерівномірний характер;
- одночасно відбуваються звернення до декількох типів ресурсів;
- інтенсивність звернення до кожного ресурсу може змінюватися в залежності від зовнішніх умов;
- через відсутність розподілу навантаження між ресурсами при піковому навантаженні обладнання не завжди дозволяє обслужити всі запити;

Можна виділити наступні класи рішень, які використовуються при побудові багато вузлових систем (рис.1):

- балансування з пропуском трафіку через один пристрій балансування;
- балансування засобами кластера;
- балансування без пропуску трафіку через один пристрій балансування.

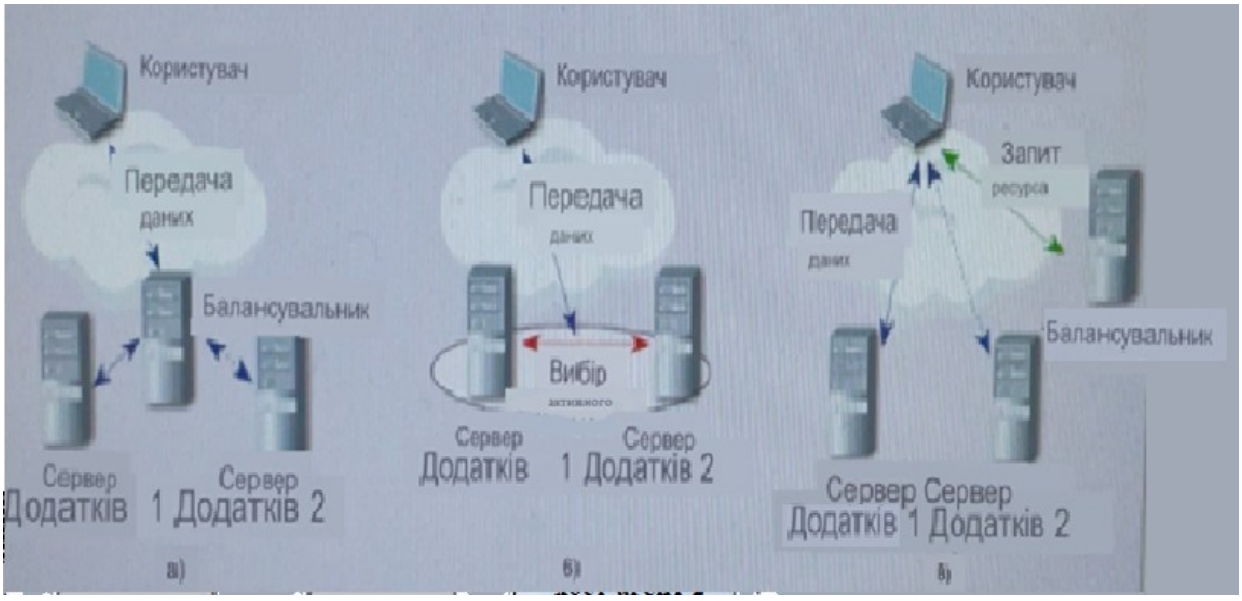

Рис. 1. Принципи балансування навантаження

Віртуальні машини дають можливість розподіляти ресурси динамічно відповідно до вимог, оптимізуючи продуктивність додатків і споживання електроенергії. Для динамічного перерозподілу ресурсів існує ряд можливостей рішення, однією з основних є жива міграція віртуальних машин. Вона дозволяє хмарним провайдерам переміщати віртуальні машини з перевантажених хостів, підтримуючи їх продуктивність при заданому SLA і динамічно консолідувати віртуальні машини на меншій кількості хостів, щоб економити електроенергію при низькому завантаженні. Використовуючи «живу» міграцію і застосовуючи онлайн-алгоритми, які дозволяють приймати рішення про міграцію в реальному часі, можна ефективно управляти хмарними ресурсами, адаптуючи розподіл ресурсів до навантажень ВМ, підтримуючи рівні їх продуктивності відповідно до SLA і знижуючи енергоспоживання інфраструктури .

Більш перспективним є підхід прийняття рішень про динамічну міграцію на основі прогнозів використання ресурсів на кілька кроків вперед. Це не тільки підвищує стабільність, оскільки міграційні дії починаються тільки коли навантаження зберігається протягом декількох тимчасових інтервалів, але також дозволяє хмарним провайдерам прогнозувати стан перевантаження до того, як це станеться. Ще одна важлива проблема полягає в тому, що жива міграція повинна виконуватися тільки в тому випадку, якщо штраф за можливі порушення SLA перевищує накладні витрати на міграцію.

Для кожної віртуальної машини існує агент ВМ, який визначає розподіл ресурсів на своїй віртуальній машині в кожному часовому інтервалі. Для кожного хоста є хост-агент, який отримує рішення про розподіл ресурсів всіх агентів ВМ і визначає остаточні параметри розподілу, вирішуючи будь-які можливі конфлікти. Він також виявляє, коли вузол перевантажений або недовантажений, і передає цю інформацію глобальному агенту. Глобальний агент ініціює рішення про міграцію віртуальної машини, переміщаючи ВМ від перевантажених або недовантажених хостів на консолідуючі хости для зменшення втрат за порушення SLA і скорочення кількості фізичних вузлів. Загальна архітектура диспетчера ресурсів і його основних компонентів показана на рис.2 [2].

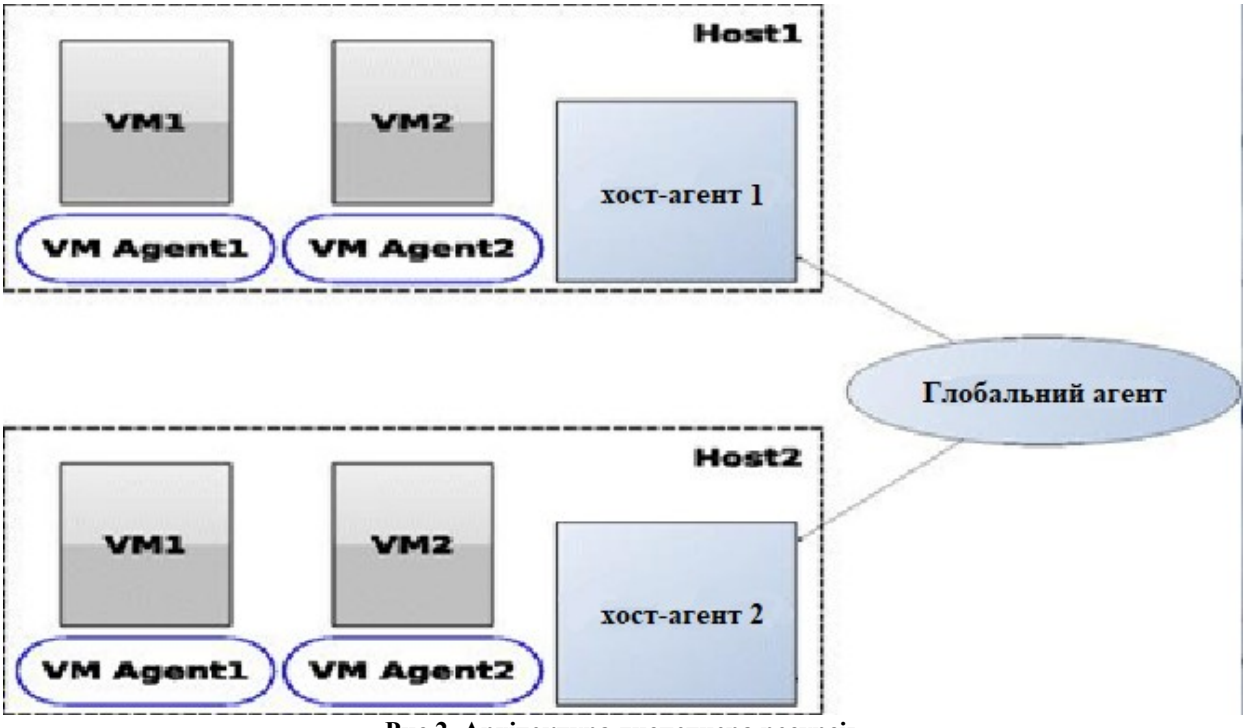

Рис.2. Архітектура диспетчера ресурсів

Щоб обчислити поточне навантаження, досліджується попит на ресурси, тобто центральний процесор (ЦП), пам’ять і пропускну здатність. ВМ надаються на хості шляхом отримання ресурсів від хоста. Навантаження на окрему віртуальну машину можна розрахувати, оскільки прогноз майбутніх потреб у ресурсах дуже важливий для хмарного ЦОД. Цей прогноз майбутнього ресурсу можна зробити за допомогою моделі линийної регресії (Linear Regression, LR). Для прогнозування використання ресурсів необхідна історія минулих потреб хосту в ресурсах. Тут

враховується потреба в ресурсі за минулу годину. Модель LR допомагає передбачити потреби в ресурсах, а потім її можна застосовувати для виявлення умов перевантаження та недовантаження. Щоб обчислити загальне навантаження на хост, можна накопичити всі наразі отримані ресурси ВМ, які працюють на хості. Якщо *m* кількість віртуальних машин, розгорнутих на *n-му* хості, тоді середнє навантаження на *n-й* хост може бути перевірено. Ці три основні параметри *і* навантаження на віртуальну машину можна розрахувати [3]:

$$VM(cpu)_{used} = \frac{\sum VM}{\sum PM}, \tag{1}$$

$$VM(bw)_{used} = \frac{\sum V.}{\sum P.}, \tag{2}$$

$$VM(ram)_{used} = \frac{\sum VI}{\sum PM}. \tag{3}$$

Використання ресурсів хостів має лінійну залежність від загального навантаження розгорнутих ВМ разом із ЦП, пропускною спроможністю пам'яті на основі згаданих вище кількох параметрів:

$$VM(util) = VM(cpu)_{used} + VM(bw)used + VM(ram)_{used} \tag{4}$$

.

ЦП відіграє найважливішу роль для надання навантаження віртуальної машини. Таким чином, можна сказати, що будь-яке навантаження на віртуальну машину прямо пропорційне потребам ЦП.

$$VM(load) = VL = \frac{\sum V.}{\sum P.}. \tag{5}$$

Отже, загальне навантаження на хост дорівнює загальній кількості ВМ, які працюють на хості. Отже, середнє навантаження на *n-й* хост фізичної машини становить:

$$PM(load) = \frac{\sum_{j=1}^{m}}{m}. \tag{6}$$

Хоча пряме підсумовування ресурсів може не гарантувати точного використання хосту. Для розрахунку використання хоста враховуються численні фактори, такі як ЦП, пам'ять і пропускна здатність, беручи продукт і об'єднуючи їх для віртуальних і фізичних машин:

$$h(u) = \frac{VM_{j(cpu)used)}}{1 \quad pm(cpu)} * \frac{VM_{j(ram)used)}}{1 \quad pm(ram)} * \frac{VM_{j(bw)used)}}{1 \quad pm(bw)}, where \tag{7}$$

Запропонований алгоритм 1 передбачає використання ЦП хоста далеко за 1 годину з 5-хвилинним інтервалом. Спосіб 1-годинного використання ЦП є важливою для збільшення короткострокового майбутнього використання. Потім він аналізує хост, чи він зараз перевантажений або недовантажений, і виконує міграцію на основі різних параметрів ресурсів фізичної машини. Алгоритм на основі LR ініціюється, щоб побудувати функцію між минулим використанням ЦП і майбутніми потребами ЦП кожного хоста за допомогою рівняння 1.

Алгоритм 1: UPLReg Алінійної Аналіз регресії з Host List передбачення корисності. Алгоритм аналізу.

Input: *Host List h*
Output: *Predict _utilization*
1 *Perquisite: k past utilization interval to approximate the prediction function;*
2 *Initialize the $в_0$ and $в_1$* by availing minimal randomized *values*;
3 For i = 1 to k interval do;
4 x_i←*Utiliz_History(i)*;
5 y_i←*Utiliz_History*(i+1);
6 $y_i = в_0 + в_1 x$;
7 /∗Go for loss calculation∗/;
8

$$\xi_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_j)^2;$$

Revise the $в_0$ value using Eq. 4.6 with the help of i samples of k;
Revise the $в_1$ value using Eq. with the help of i samples of k;
End For;
// Using regression function;

Predict_utilization = $в_0$ + $в_1$ ∗ Current_total_Utilization(h);

return Predict_utilization.

В Алгоритмі 1 спочатку прогнозовані малі випадкові параметри коефіцієнта k і $в_1$ на наступному етапі функція передбачення визначається на основі k попередньо використаної історії в хості h. Від рядків 3 до 11 з 1 моделі LR обчислює відмінності між фактичним і очікуваним навантаженням у кожній точці даних, і тут це може бути наступне значення навантаження, яке є фактичним для попередньої точки даних (рядки 4 і 5 Алгоритму 1). Очікуване використання оцінюється на основі поточного використання та $в_0$ і $в_1$ (у рядку 6 з 1). Для розрахунку суми збитків квадрата залишків використовується загальна і минула точка даних (рядок 7 і 8 з 1). $в_0$ і $в_1$ оновлюються в рядках 9 і 10, щоб мінімізувати різницю між ними фактичне і прогнозоване використання. Нарешті, Predict_utilization використовується для прогнозування використання на основі останнього загального використання. У 2 Migrate Live_VM використовуються для ініціювання двох різних черг, перша – це недовантажений пул хостів, що складається з ідентифікатора, який є недостатньо завантаженим, а друга – список ВМ, які потребують міграції, оскільки їх хост стає перевантаженим.

Алгоритм 2: *Migrat eLive_V Ms*

Input: *host List h*
Output: *$queue_1$: V Mlistthatareneedtomigrate, $queue_2$: underloadedhostwhereV Mscanbemigrated*
1 For each *host list* do;
2 If(*Detect _over load (host)* is *true*) ;
3 Select VM with host *id* (*V M , h*) to migrate from the corresponding host and put it into the *$queue_1$*;
4 ELSE If*(Detect _under load (host)* is *true*));
5 Select host id h and put back into the *$queue_2$*;

```
6 queue2 := h;
7 End For.
```

Алгоритм 3 використовується тут для виявлення перевантаженого хосту *h* його повернуте значення є істинним, якщо хост здається перевантаженим, а повернуте значення є хибним, доки на хості *h*. Тут хост вважається перевантаженим, якщо поточне використання ресурсів у хості становить 85% (значення було знайдено емпіричними вимірюваннями та експериментальним аналізом) від загального використання протягом початкової 1 години, після чого хост стає перевантаженим. Крім того, згідно з алгоритмом 1, якщо отриманоо значення pr ed i ct ed _ut il більше, ніж поточне загальне значення використання, то конкретний хост стає перевантаженим.

Алгоритм 3: Виявлення вихідного перевантаженого хоста

```
Input: host h
Output: 1 if host is overloaded else 0
1  If (utliz_history_lengt h < 12);
2  If (current _total _utilz(host) > 0.85 ∗ (total_utiliz(host));
3  return TRUE;
4  else;
5  return FALSE;
6  end if;
7  end if;
8  else;
9  predicted_utill := UPL Reg A(host );
10 If (predicted _utill > curent_total_utilz(host));
11 return TRUE;
12 Else;
13 return FALSE;
14 end if;
15 end else
```

Висновки

1. Хмарне обчислення вимагає розгортання середовища хостів та ВМ таким чином, щоб значна кількість користувачів могла одночасно отримувати хмарні послуги в ЦОД Call Center без порушенн SLA. Це головна мета хмарних обчислень в ЦОД Call Center.
2. Необхідно враховувати кілька параметрів, таких як ресурси, пам’ять, кількість мережних компонентів для ЦОД Call Center. Більшість параметрів безпосередньо впливають на продуктивність будь-якого підходу до балансування навантаження.
3. Інфраструктура хмарних обчислень розподіляється та розширюється в різних географічних місцях корпоративних мереж зі ЦОД Call Center, починаючі з глобального рівня.

Література

1. Das R., Kephart J.O., Lefurgy C., Tesauro G., Levine D.W., Chan H. Autonomic multi-agent management of power and performance in data centers // Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track. International foundation for autonomous agents and multiagent systems. 2018. P. 107–114.

2. Farahnakian F., Liljeberg P., Plosila J. LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers // Proceedings of the 39th euromicro conference on software engineering and advanced applications. IEEE. 2013. P. 357–364.

3. Sajitha A.V., Subhajini A.C. Dynamic VM consolidation enhancement for designing and evaluation of energy efficiency in green data centers using regression analysis // Int J Eng Technol. Vol. 7 (3.6). P.179–186.

4. Лук'янов О.І., Сабурова С.О. Аналіз методів моніторингу показників якості функціонування CALL CENTER CRM // Матеріали Міжнародної науково-технічної конференції «Інформаційно-комунікаційні технології та кібербезпека» (ІКТК-2024). Харків. ХНУРЕ. 2024. С. 68-71.