УДК 004.4

ДО ЗАДАЧІ ОПТИМІЗАЦІЇ ЗАВАНТАЖЕНОСТІ РЕСУРСІВ ОБЧИСЛЮВАЛЬНОГО КЛАСТЕРА З ВУЗЛАМИ У ВИГЛЯДІ ВІРТУАЛЬНИХ МАШИН

С.Д. Погорілий, І.В. Білоконь

Київський національний університет імені Тараса Шевченка,
01601, вул. Володимирська, 64, Київ, Україна,
E-mail: sdp@univ.net.ua, deimos@univ.net.ua

Виконано формалізацію задачі оптимізації конфігурації обчислювального кластера, вузлами якого є віртуальні машини. Запропоновано підхід, що грунтується на застосуванні генетичних алгоритмів, які вимагають формування фітнес-функції відповідно до сформульованої задачі. Наведено результати низки експериментів для підтвердження доцільності застосування запропонованого підходу.

Formalization of configuration optimization problem for computing cluster with virtual machine nodes was performed. Approach based on genetic algorithm applying which require fitness function implementation according to problem statement was proposed. Several experimental results were shown in order to justify proposed approach.

Вступ

Різноманіття паралельних обчислювальних задач та прагнення до оптимізації використання наявних обчислювальних ресурсів призвели до необхідності побудови кластерів із гнучкою архітектурою. Величезна кількість програмного забезпечення, реалізованого для паралельних обчислень на традиційних серверних архітектурах та персональних комп'ютерах (ПК), призвела до необхідності пошуку можливостей побудови гнучких обчислювальних систем на базі саме таких обчислювальних вузлів. Серед запропонованих підходів варто виділити створення обчислювальних кластерів на базі наявних апаратних обчислювальних ресурсів (ПК або серверів) з можливістю регулювання кількості обчислювальних вузлів у залежності від потреб, часу доби тощо [1–3]. Розвитком цих принципів є побудова обчислювальних кластерів на базі віртуальних машин [4]. Віртуалізація дозволяє використовувати апаратні можливості комп'ютера більш раціонально та динамічно змінювати конфігурацію обчислювальних ресурсів віртуального сервера без внесення змін до апаратного забезпечення фізичної платформи.

Можливість програмної реконфігурації ресурсів обчислювального кластера, побудованого на базі віртуальних машин (ВМ), дозволяє проводити різноманітні дослідження конфігурацій, наприклад, пошук оптимальних конфігурацій для певних задач. В роботі [5] показана можливість впливу на продуктивність роботи програмних реалізацій паралельних алгоритмів шляхом варіювання конфігурацій кластера, побудованого на ВМ платформи Microsoft Hyper-V R2 [6]. Результати досліджень [5] обгрунтовують актуальність розв'язання задач пошуку оптимальних конфігурацій.

Оскільки задача оптимізації передбачає мінімізацію (максимізацію) певного функціонала, у випадку конфігурацій кластерної системи необхідно визначити критерії, за якими їх порівнювати. Продуктивність роботи кластера на кожній конкретній задачі залежить як від його апаратно-програмної конфігурації, так і від реалізації самої задачі та її вхідних даних. Зазвичай час роботи задачі становить дні, тижні, або навіть місяці. Деякі задачі можуть взагалі не мати логічного завершення, а лише практично нескінченно покращують точність результатів. Тому при оцінці якості конфігурацій не доцільно використовувати дані про весь життєвий цикл програми, що автоматично означає неможливість врахування такого показника як час виконання програми. Таким чином, потреба у об'єктивних метриках, які б характеризували якість тих чи інших конфігурацій для відповідних задач, є очевидною. У цій роботі запропоновано підхід до побудови метрики якості конфігурацій та наведено результати досліджень продуктивності обчислювального кластера, вузлами якого є ВМ на платформі Microsoft Hyper-V R2, в залежності від його конфігурацій для ілюстрації ефективності запропонованого підходу.

Задача оптимізації конфігурації обчислювального кластера

Динамічна реконфігурація ресурсів обчислювального кластера. Експлуатація застосувань і ОС ВМ майже не відрізняється у порівнянні із фізичними ПК. Але зміна конфігурацій, реалізація відмовостійкості, масштабованість систем при використанні ВМ значно спрощує роботу системних адміністраторів. Навідміну від ПК, операції зміни апаратних конфігурацій віртуального обладнання ВМ зводяться до виклику функцій

прикладного програмного інтерфейсу (API) платформи віртуалізації. Функції API платформи Microsoft Hyper-V R2 [4, 7] надають цілу низку засобів оперуванням ресурсами ВМ, основними з яких є наступні.

1. Модифікація віртуального процесора ВМ. Впливає на кількість паралельних процесорів, доступних програмам і операційній системі (ОС) ВМ і фактично розв'язується зміною кількості віртуальних процесорів (1-4 шт.).

2. Модифікація віртуальної оперативної пам'яті (ОП). Впливає на обсяг ОП, доступний програмам і ОС ВМ:

– зміна типу ОП (статична, динамічна – ВМ в процесі роботи відповідно до потреб займає чи вивільнює частину ОП сервера віртуальних машин (СВМ), на якому знаходиться);

– зміна обсягу ОП (у випадку статичної ОП – фактичний обсяг ОП, доступний ВМ, у випадку динамічної ОП – обсяг ОП, необхідний для запуску ВМ).

3. Модифікація розташування ВМ. Впливає на швидкість і затримки мережевого з'єднання між ВМ (ВМ, розташовані в межах одного СВМ, поєднуються мережею з пропускною здатністю 10 Гбіт/с і значно меншими затримками); з іншого боку, розташування декількох ВМ на одному СВМ обмежує обсяг доступної ОП, а також може призвести до змагання за ресурси центрального процесора (ЦП) СВМ (оскільки між ядрами ЦП СВМ і процесорами ВМ немає взаємно однозначної відповідності і процесорний час надається відповідно кількості віртуальних процесорів ВМ і встановленим пріоритетам):

– швидка міграція (перенесення ВМ з одного СВМ на інший із тимчасовим призупиненням роботи ВМ);

– жива міграція (перенесення ВМ з одного СВМ на інший без призупинення роботи ВМ).

Розглядаючи ВМ як вузли обчислювального кластера необхідно звертати увагу на вплив наведених операцій на хід виконання завдань. Наприклад, операції (1) і (2) потребують вимкнення ВМ для встановлення параметрів. Таким чином, у випадку наявності активних завдань, їх доведеться перезапускати після виконання зазначених дій.

Це не всі можливі операції впливу на ресурси ВМ, але навіть за допомогою наведених засобів можна реалізувати величезну кількість різноманітних конфігурацій. Лише кількість модифікацій розташування можна оцінити як m^n, де m – кількість ВМ, а n – кількість СВМ. Таким чином, будь-яка задача пошуку на множині можливих конфігурацій буде *NP* повною задачею у k-вимірному просторі, де k – кількість параметрів ВМ, що підлягають реконфігурації.

Постановка задачі. Перш ніж перейти до формулювання задачі оптимізації необхідно визначити, які показники за яких умов підлягають покращенню. Уявимо традиційний обчислювальний кластер, завантаженість якого забезпечується алгоритмом планувальника (диспетчера) завдань. Внаслідок роботи алгоритму, до виконання будуть допущені завдання із черги відповідно до їх налаштувань (пріоритети, необхідна кількість вузлів/процесорів/ядер/ОП тощо), причому в черзі залишаться лише ті завдання, для яких не лишилося такої кількості вільних обчислювальних ресурсів, яка відповідає їх мінімальним потребам. Таким чином, з точки зору планувальника, ресурси кластера завантажені найоптимальнішим шляхом. Але з точки зору конкретних задач це може виявитися не так, оскільки програмні реалізації паралельних алгоритмів висувають цілу низку вимог до міжпроцесного, міжвузлового і міжпотокового зв'язку. Тому подальшу оптимізацію можливо реалізувати лише на більш низькому рівні.

Хоча застосування віртуалізації не може дати покращення продуктивності апаратних засобів, цей підхід дозволить збільшити пропускну здатність обчислювального кластера (кількість виконаних завдань або кількість ітерацій нескінченних завдань). Цього можна досягти за рахунок «ущільнення» завантаженості апаратних засобів. Таке ущільнення реалізується шляхом перерозподілу навантажень по СВМ, або створення більш сприятливих умов для завдань. Наприклад, 4-ядерний ЦП на певному СВМ завантажено 4 процесами задач так, що його загальна завантаженість сягає лише 75 % із 100 % можливих. Це відбувається внаслідок очікування ресурсів, результатів роботи інших процесів тощо. Шляхом підбору певної конфігурації кластера ми можемо підвищити цей показник за рахунок зменшення часу очікування ресурсів, або якщо це неможливо, розмістити на даному СВМ додатковий вузол, надавши йому 25 % процесорного часу СВМ (наприклад, створивши ВМ з одним віртуальним процесором). Таким чином ми або пришвидшимо роботу встановлених завдань, або створимо ресурси для інших завдань, що містяться в черзі. Отже, показником якості конфігурації обчислювального кластера будемо вважати його пропускну здатність. Оскільки попередньо виміряти цей показник неможливо, будемо намагатися передбачити його поведінку, досліджуючи «ущільнення» завантаженості ЦП СВМ.

Нехай маємо комп'ютерний клас з N ПК. Сформулюємо у загальному вигляді задачу оптимізації завантаженості ресурсів обчислювального кластера, враховуючи лише операції керування ресурсами ВМ, для чого виконаємо наступну формалізацію.

Нехай P_i, i=1,2...N – множина віртуальних машин на СВМ з індексом i. Тоді множина всіх наявних віртуальних машин становитиме об'єднання множин P_i:

$$V = \{v_j\} = \bigcup_i P_i, j = 1,2...W . \quad (1)$$

Кількість ВМ на СВМ з індексом *i*:

$$W_i = \sum_j X_{ij}, i = 1,2...N \,. \tag{2}$$

Координати ВМ:

$$X_{ij} = \begin{cases} 1, v_j \in P_i \\ 0, v_j \notin P_i \end{cases}, X_{ij} \in A_p, p \in \{1,2...n\}. \tag{3}$$

Кількість паралельних обчислювальних ядер на фізичній машині з індексом *i*: C_i, i=1,2...*N*.
Кількість віртуальних процесорів *j*-ї ВМ:

$$G_j \in A_i, i \in \{1,2,...,n\}. \tag{4}$$

Множини значень параметрів ВМ: A_1.. A_n.
Множина всіх можливих конфігурацій будь-якої ВМ:

$$A = A_1 \times A_2 \times ... \times A_n \,. \tag{5}$$

Конфігурація обчислювальної системи:

$$z \in \prod_j A_j \,. \tag{6}$$

Завантаженість *k*-го ядра процесора *i*-го СВМ.

$$L_{ik} = L_{ik}(z), k = 1,2...C_i \,. \tag{7}$$

Завантаженість *m*-го процесора *j*-ї ВМ:

$$M_{jm} = M_{jm}(z), m = 1..G_j \,. \tag{8}$$

Сумарна завантаженість процесорів всіх СВМ:

$$F(z) = \sum_i \sum_k L_{ik} \,. \tag{9}$$

Сумарна завантаженість процесорів усіх ВМ на СВМ з індексом *i*:

$$H_i(z) = \sum_j \sum_m M_{jm} * X_{ij} \,. \tag{10}$$

Тут під терміном «завантаженість» розуміємо середню завантаженість за час проведення експеримента (вимірів). За основну метрику візьмемо величину сумарної завантаженості ЦП $F(z)$. Її зростання характеризуватиме покращення умов для поточних задач і, відповідно, менший час очікування ресурсів. Розділення гостьових ОС і відсутність взаємно однозначної відповідності між ЦП ВМ і ЦП СВМ призводить до необхідності накладання додаткових умов на те, що сумарна завантаженість всіх ядер (процесорів) всіх ВМ на СВМ не повинна перевищувати завантаженість всіх ядер ЦП СВМ, бо інакше буде відбуватися змагання між ВМ за ресурси (наприклад, 100 % завантаження ЦП ВМ може призвести до такого самого загального завантаження ЦП СВМ; цей показник збережеться і при додаванні на даний СВМ інших ВМ). Таким чином, отримуємо задачу оптимізації завантаженості ресурсів кластера за допомогою керування ресурсів ВМ:

Знайти *z* за умови:

$$\begin{cases} F(z) \to max \\ (\forall i \in \{1,2,...N\}) \left[H_i(z) \le \sum_k L_{ik} \right] \\ N = const \end{cases}. \tag{11}$$

Ця задача є задачею багатокритеріальної оптимізації і може припускати різних підходів до її розв'язання. Оскільки пошук оптимальних значень відбувається у багатовимірному просторі, доцільно застосувати генетичні алгоритми, в яких особинами популяцій є конфігурації кластера. При застосуванні генетичних алгоритмів особливістю цієї задачі є значна тривалість визначення значень функцій (тому що фактично це означає проведення експерименту по зміні конфігурації та вимірюванню показників, таких як

завантаженість процесора). Як відомо, генетичний підхід потребує великого різноманіття особин популяцій, а отже великої кількості обрахунків значень фітнес-функції, що відповідно означає неможливість побудови фітнес-функцій для визначення якості конфігурацій, базуючись на переборі можливих конфігурацій (незалежно від алгоритму перебору) та вимірюванні значень $F(z)$. Тому слід будувати певні моделі, що дозволять обчислювати шукані фітнес-функції без проведення фактичних вимірів.

Зазвичай набір паралельних програм, що працюють на кластері обмежений. Тому першим кроком до побудови фітнес функції може бути попереднє вимірювання шуканих значень для кожної програми окремо і застосування отриманих результатів для сукупності завдань кластера. Незважаючи на значно меншу кількість варіантів конфігурацій, порівняно із множиною завдань (кількість проведених експериментів становитиме суму кількостей для окремих завдань, а не добуток), перебір всіх можливих комбінацій для кожного завдання окремо все одно не є здійсненним в загальному випадку. Як один із можливих підходів до розв'язку в роботі запропоновано проведення вимірювань лише для деяких ключових варіантів конфігурацій, що дозволять передбачити поведінку фітнес-функції для інших варіантів.

Ключові конфігурації та метрики. При штатній роботі обчислювального кластера можна вважати, що єдиним чинником, що впливає на продуктивність виконання обчислень є кількість і якість доступних процесам завдань обчислювальних ресурсів (швидкість роботи ЦП, обсяг і швидкість доступу до ОП, швидкість і затримки мережевого з'єднання). Вищенаведені операції з реконфігурації ВМ дають можливість корегувати ці параметри в певних межах як для кожної ВМ, так і для їх сукупності. Розглядаючи конфігурації кластера як сукупність конфігурацій його вузлів можна зробити наступні доволі обґрунтовані припущення про ступінь можливого впливу якості вузлів на розподіл обчислювальних ресурсів між процесами завдань і, відповідно, на покращення чи погіршення результуючої продуктивності обчислень. Так розміщення однопроцесорних ВМ на СВМ 1 до 1 (1 ВМ на 1 СВМ) створить вузли із максимальною кількістю доступної процесу задачі ОП, а таке саме розташування ВМ із 4 процесорами дозволить запустити більшу кількість завдань, але їх процеси будуть ділити доступну ОП між собою. Відповідно перша конфігурація буде більш сприятливою для задач, процеси яких потребують якомога більшого обсягу ОП, а друга дозволить одночасно і ефективно працювати більшій кількості завдань із меншими потребами в ОП. З іншого боку, розміщення декількох ВМ на 1 СВМ є вигіднішим більш вимогливим до затримок мережевого з'єднання і одночасно менш вимогливим до процесорних ресурсів завданням.

Отже, маючи інформацію про найбільш критичні обчислювальні ресурси для відповідних завдань, можна принаймні передбачити властивості, які повинні мати оптимальні конфігурації для цих завдань. Базуючись на таких міркуваннях будемо обирати ключові конфігурації, які б дозволили певним чином класифікувати завдання за характером потреб у ресурсах.

Розглянемо конфігурацію обчислювального кластера як розподіл процесів завдання по СВМ. Ключовими варіантами розподілу будемо вважати наступні.

1. Кожен процес завдання розташований на СВМ так, що крім нього на даному СВМ не виконується жодного іншого завдання, тобто 1 процес на 1 СВМ. Така конфігурація дозволить дослідити як впливають на продуктивність максимально можливі обсяги доступної ОП. Позначимо цю конфігурацію К1.
2. Всі процеси завдання розташовані на одному і тому самому СВМ. Ця конфігурація буде сприятливою для завдань, що потребують малих затримок і великої пропускної здатності мережевого з'єднання і водночас не потребують значної кількості ресурсів ЦП. Позначимо конфігурацію як К2.
3. Кількість процесів на СВМ дорівнює кількості ядер ЦП СВМ. Причому, якщо кількість процесів не ділиться націло на кількість ядер, то один СВМ буде містити кількість процесів, що дорівнює остачі від ділення (наприклад, якщо кількість процесів 14, а кожен СВМ має 4-ядерний процесор, то 3 СВМ міститимуть по 4 процеси, а четвертий – 2 процеси). Ця конфігурація є стандартною конфігурацією вузлів кластера. При кількості процесів завдання меншій або рівній кількості ядер ЦП СВМ вона збігається із К2. Позначимо конфігурацію як К3.

Для зменшення часу проведення вимірювань та кількості експериментів використаємо наступні припущення.

1. Всі процеси завдання однакові.
2. Кожен процес завдання послідовний.
3. Завдання може мати будь-яку парну кількість процесів.
4. Всі вузли ВМ, що є вузлами кластера містяться на однакових СВМ. СВМ не містять інших ВМ, окрім вузлів кластера.
5. Як параметри ВМ використовуємо її координати та кількість процесорів, а також кількість ВМ. ОП вважаємо динамічною, що може бути змінена в межах доступної фізичної ОП СВМ.

У процесі роботи завдання на кластері існує можливість вимірювання лічильників продуктивності (performance counters), таких як загальна завантаженість процесора, обсяг зайнятої ОП, використання файлу підкачки, обсяг переданих і отриманих по мережі даних, кількість відправлених і прийнятих мережевих пакетів тощо. Значення таких параметрів визначаються в певні моменти часу і не характеризуються стабільністю і повторюваністю, тому використовувати їх безпосередньо не варто. Згідно вищесформульованої задачі, в цій

роботі використано виміри завантаженості процесорів обчислювальних вузлів для отримання характеристик якості конфігурацій. Числові значення характеристик (фітнес-функція) формуються наступним чином:

$$\varphi(z) = \frac{\sum_i \overline{X}_i}{t}. \qquad (12)$$

Тут $\overline{X}_i$ – середнє значення завантаженості ЦП обчислювальних вузлів, отримане в певні моменти часу, t – проміжок часу, за який проведено вимірювання. Дана величина фактично є середньою завантаженістю ресурсів ЦП за час проведення експеримента.

Передбачення поведінки фітнес-функції для інших конфігурацій можна виконати виходячи із наступних міркувань. Зважаючи на ідентичність процесів завдання, перебіг обчислень при довільній припустимій конфігурації обчислювального кластера буде відбуватися аналогічно перебігу обчислень при одній із ключових конфігурацій із такою самою мірою доступності критичного обчислювального ресурсу. Наприклад, конфігурація по 2 процеси на СВМ за умов достатнього обсягу ОП буде впливати на продуктивність так само, як і конфігурація К1, а будь-яке розміщення всіх 8 процесів завдання на 1 СВМ буде відповідати конфігурації К2.

Методика вимірювань

Автоматизація експеримента. Як було зазначено в розділі постановки задачі, реконфігурацію обчислювальної системи і проведення вимірювань необхідно виконувати окремо для кожного завдання. Також бажано, щоб цей процес залишився поза увагою користувачів і ніяк не впливав на їх процес роботи. Оскільки планування черги кластера основане на даних про завдання, введених користувачем під час його налаштування, після прийняття планувальником черги рішення про надання обчислювальних ресурсів неможливо вплинути на властивості завдання (необхідна кількість вузлів чи ядер, час виконання, командний рядок тощо), не скасовуючи його. Процес реконфігурації може потребувати встановлення завдань на конкретні вузли та модифікацію кількості вузлів, що може не відповідати вихідним параметрам завдання. Тому необхідно виконувати дії над кластером після налаштування завдання користувачем, але до його запуску. В кластері Microsoft HPC Server 2008 R2 це можна здійснити за допомогою механізму фільтрів [8].

Життєвий цикл завдання на кластері Microsoft HPC Server 2008 R2 а також послідовність спрацьовування фільтрів показано на рис. 1.

Фільтр являє собою застосування, що приймає на вхід шлях до файлу формату XML (Extensible Markup Language), який містить опис завдання, а в результаті роботи повертає певне числове значення, яке використовується планувальником для прийняття рішення щодо подальшої долі завдання (рис. 2). Існує 2 типи

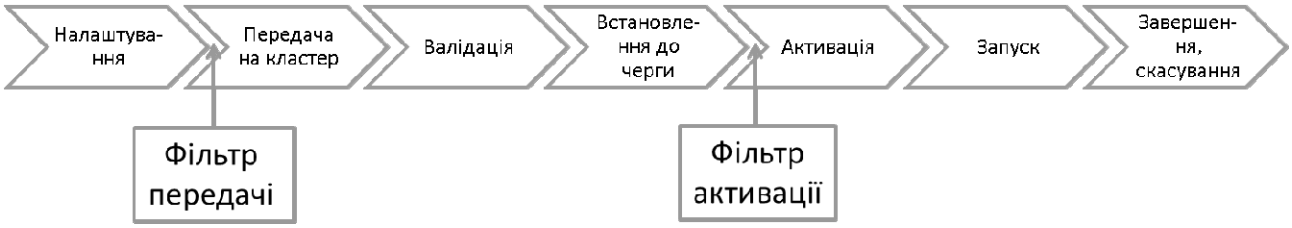


Рис. 1. Життєвий цикл завдання на кластері

фільтрів: фільтр передачі та фільтр активації.

Фільтр передачі (рис. 2, а) виконується після налаштування користувачем параметрів завдання. В результаті роботи ці параметри можуть бути змінені логікою фільтра. Значення, що повертаються застосуванням фільтра, інтерпретуються кластером наступним чином: 0 ==> залишити завдання без змін; 1 ==> завдання модифіковано; будь-яке інше значення ==> скасувати завдання.

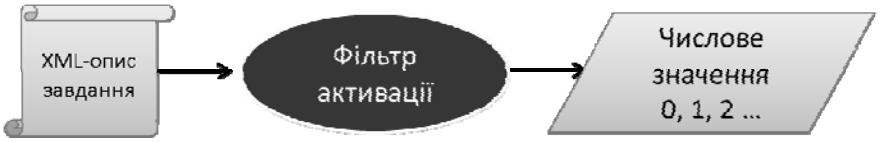
Фільтр активації (рис. 2, б) виконується після прийняття алгоритмом планувальника черги рішення про надання ресурсів завданню. Цей тип фільтра не дозволяє здійснювати жодних модифікацій завдання, а лише може бути використаний для затримання виконання завдання чи відхилення завдання.

Із вищеописаних міркувань випливає необхідність використання саме фільтру передачі для експериментів по реконфігурації кластера. Принцип побудови фільтра для автоматизованої системи вимірювання показників продуктивності мовою C# у вигляді консольного застосування наведено далі.

```
class Program
   {
      static int Main(string[] args)
      {
         using (IScheduler scheduler = new Scheduler())
         {
            scheduler.Connect("HeadNodeName");
```

```
            ISchedulerJob job = scheduler.CreateJob();
            job.RestoreFromXml(args[0]);
            //виконуємо експерименти
        }
        return 0;
    }
}
```

а



б

Рис. 2. Фільтри Microsoft HPC Server 2008 R2: а – фільтр передачі; б – фільтр активації

Алгоритм експеримента в загальному вигляді можна сформулювати наступним чином.

1. Отримуємо дані, введені користувачем.
2. Формуємо набір ключових конфігурацій із врахуванням встановлених користувачем параметрів завдання.
3. Для кожної ключової конфігурації виконуємо дії 4–9.
4. Реалізуємо розташування ВМ по СВМ шляхом швидкої або живої міграції.
5. Змінюємо кількість процесорів ВМ, якщо необхідно. Для цього вимикаємо відповідні ВМ, робимо зміни, вмикаємо ВМ.
6. Очікуємо, доки кластер встановить контакт із всіма вузлами.
7. Формуємо завдання, ініціалізуючи його даними, введеними користувачем. Вносимо необхідні відповідно до конфігурації зміни в параметри завдання (кількість ядер для завдання, вузли, на яких виконувати завдання, час виконання завдання тощо).
8. Передаємо завдання на кластер і очікуємо його запуск.
9. Поки завдання працює вимірюємо показники продуктивності.

Результати експериментів. Для перевірки поведінки значень запропонованої фітнес-функції у залежності від конфігурацій обчислювального кластера було проведено низку експериментів. Як тестову задачу було використано пакет LINPACK [9] з розмірностями задачі N = 4608, 15936, 31872, та параметром NB = 192. Збільшення розмірності задачі відповідає збільшенню необхідного для роботи програми обсягу ОП на кожному вузлі. Загальний обсяг ділиться між процесами завдання. Збір даних про завантаженість процесора відбувався протягом періоду роботи завдання із моменту запуску до моменту завершення. Для проведення вимірювань було створено автоматизовану систему, що є фільтром передачі, який виконує динамічну реконфігурацію обчислювального кластера та збір експериментальних даних.

Дослідження проводилися на базі комп'ютерного класу з 8 ПК наступної конфігурації:

Intel Core 2 Quad CPU Q6600 2.4 GHz.

8 GB RAM.

Gigabit Ethernet NIC.

Windows Server 2008 R2 Enterprise.

Обчислювальний кластер побудовано з використанням пакету Microsoft HPC Server 2008 R2 [10]. Враховуючи, що кількість ядер вказаного процесора становить 4, кількісний розподіл процесів по СВМ відповідно до їх загальної кількості та конфігурацій наведено у таблиці.

В конфігураціях К1 та К3 кількість процесів на СВМ відповідає кількості віртуальних процесорів ВМ, розташованої на відповідному СВМ. Оскільки максимальна кількість процесорів ВМ дорівнює 4, конфігурації

К2 для більшої кількості процесів реалізовані шляхом розташування всіх ВМ відповідної конфігурації К3 на одному СВМ.

На рис. 1 показано діаграми залежностей фітнес-функції φ та результатів роботи тесту LINPACK (продуктивності) від кількості процесів, на яких було запущено пакет, при різних конфігураціях обчислювального кластера. Відсутність значень означає неможливість реалізації конфігурації при відповідних параметрах (в даному випадку через нестачу ОП).

Наведені діаграми свідчать, що значення φ досить добре визначає найкращу конфігурацію. Поведінка значень функції дозволяє визначити чутливість задачі до зміни параметрів. Із діаграм випливає, що при невеликих розмірах вхідних даних найкращу продуктивність LINPACK показує при конфігураціях К3 незалежно від кількості процесів. Збільшення потреб в ОП призводить до «виграшу» конфігурацій із меншою щільністю розподілу процесів по СВМ, а також неможливості реалізації деяких конфігурацій взагалі.

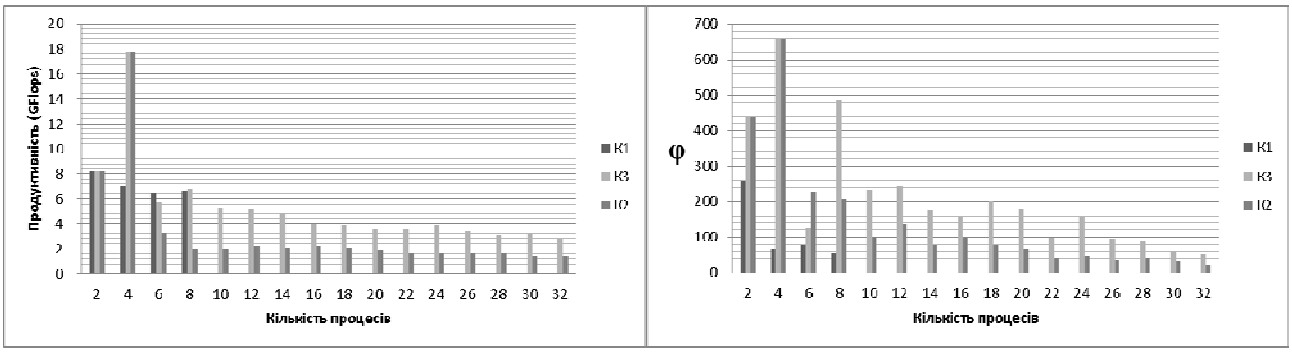
Таблиця

Конфігу-рація	*К1*								*К3*								*К2*							
Номер СВМ / Кількість процесів	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
2	1	1							2								2							
4	1	1	1	1					4								4							
6	1	1	1	1	1	1			4	2							6							
8	1	1	1	1	1	1	1	1	4	4							8							
10									4	4	2						10							
12									4	4	4						12							
14									4	4	4	2					14							
16									4	4	4	4					16							
18									4	4	4	4	2				18							
20									4	4	4	4	4				20							
22									4	4	4	4	4	2			22							
24									4	4	4	4	4	4			24							
26									4	4	4	4	4	4	2		26							
28									4	4	4	4	4	4	4		28							
30									4	4	4	4	4	4	4	2	30							
32									4	4	4	4	4	4	4	4	32							

Невідповідність у деяких випадках поведінки значень φ та продуктивності можна пояснити тим, що згідно її визначення, вона надає перевагу конфігураціям з найбільш рівномірною завантаженістю процесорів вузлів.

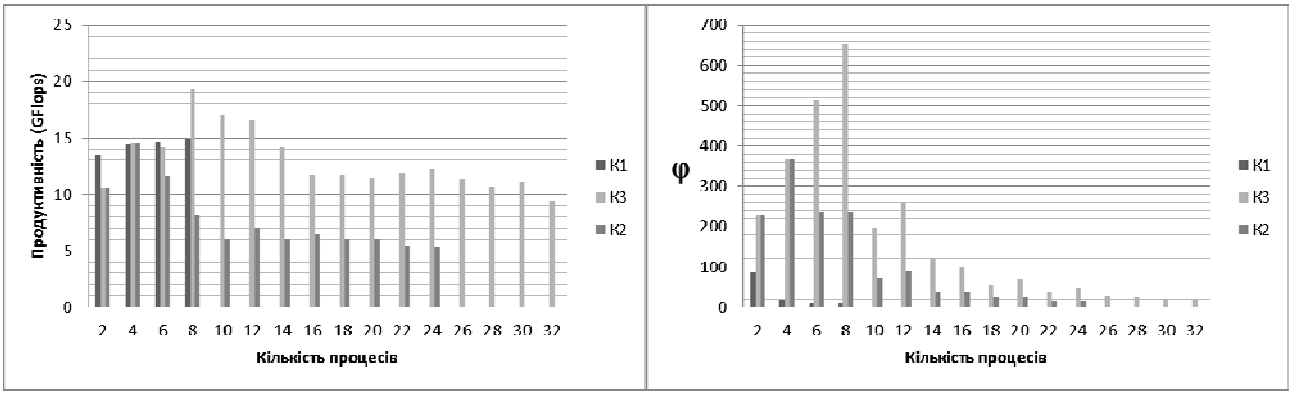
Недоліком запропонованого підходу є неврахування методу передачі повідомлень між процесами. Це пояснює перевагу конфігурацій К3 над К1 для більшості випадків, оскільки за замовчанням LINPACK використовує кільцеву передачу повідомлень (0→1, 1→2 …, де 0, 1, 2 – номер процесу). Оскільки процеси завдання зазвичай впорядковані за послідовністю зазначення вузлів при налаштуванні, більшість актів передачі повідомлень відбувається всередині одного вузла без фактичного використання мережі при конфігураціях К3. Тобто наведений у роботі набір ключових конфігурацій та виключно метрики завантаженості процесора в деяких випадках не дозволяють сформувати найбільш вигідні групування процесів за методом передачі повідомлень. Для завдань із специфічною топологією мережевих зв'язків між процесами, необхідні додаткові дослідження метрик використання мережі і, можливо, формування додаткового набору ключових конфігурацій.

Наведені результати отримані шляхом вимірювань протягом всього життєвого циклу завдання для отримання числових значень продуктивності роботи пакету LINPACK. Такий підхід неможливо застосувати для випадку реальних обчислювальних завдань через величину часу їх роботи. При дослідженні довго працюючих завдань слід проводити вимірювання протягом деякого невеликого проміжку часу. Це буде предметом подальших досліджень.



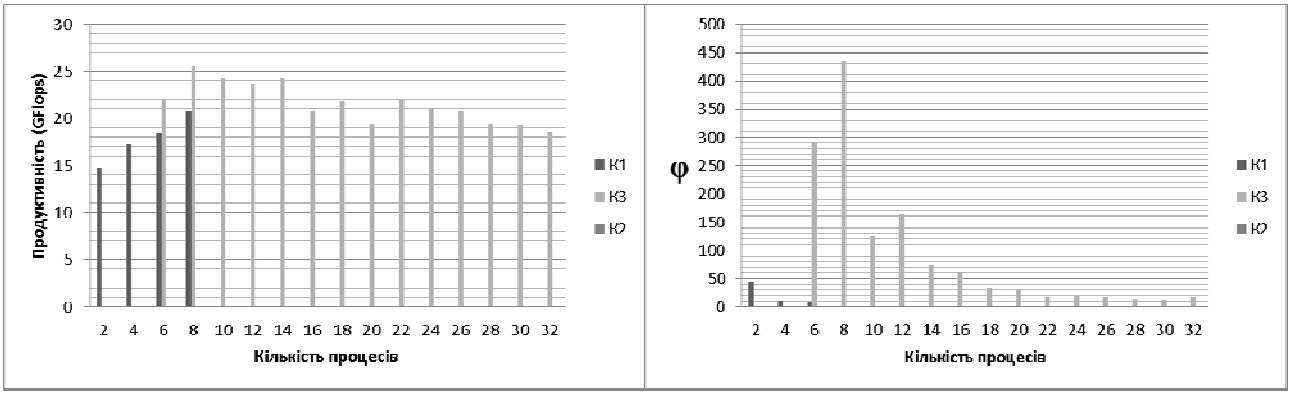
а

N=4608



б

N=15936



в

N=31872

Рис. 3. Залежність продуктивності роботи LINPACK різних розмірностей вхідних даних (*N*) та фітнес-функції від кількості процесів при різних конфігураціях

Висновки

Виконано формалізацію і формулювання задачі оптимізації конфігурації обчислювального кластера, вузлами якого є ВМ.

Запропоновано метод побудови фітнес-функції відповідно до формулювання задачі.

Проведено низку експериментів, результати яких підтвердили актуальність використання запропонованої фітнес-функції.

1. *Погорілий С.Д., Бойко Ю.В., Грязнов Д.Б.* [*та інші*] Концепція створення гнучких гомогенних архітектур кластерних систем // Матеріали 6 Міжнар. науково-практичної конф. з програмування «УкрПРОГ–2008» 27-29 травня 2008 р, Україна, Київ. – 2008. – № 2-3. – С. 84–90.
2. *Білоконь І., Грязнов Д., Мар'яновський В.* Створення захищеного обчислювального windows-кластеру на платформі desktop PC // Вісник Київського національного університету імені Тараса Шевченка. Серія: радіофізика та електроніка. Вип. 12. — К., 2009.
3. *Windows* HPC Server 2008 R2 Technical Library. Доступ до документів (21.11.2011): http://technet.microsoft.com/en-us/library/ee783547%28WS.10%29.aspx
4. *Білоконь І., Грязнов Д., Погорілий С.* Побудова динамічно реконфігурованої обчислювальної архітектури з використанням технологій віртуалізації // Вісник Київського національного університету імені Тараса Шевченка. Серія: радіофізика та електроніка. Вип. 14. — К., 2010.
5. *Білоконь І., Грязнов Д., Погорілий С.* Дослідження впливу конфігурації обчислювальної системи на продуктивність програмних реалізацій паралельних алгоритмів // Праці VII Міжнар. конф. «Електроніка та прикладна фізика» 19-22 жовтня 2011 р. Україна, Київ.
6. *Microsoft* Hyper-V. Доступ до документів (14.12.2011): http://technet.microsoft.com/en-us/library/cc753637%28WS.10%29.aspx
7. *Hyper-V* WMI Provider. Доступ до документу (14.12.2011): http://msdn.microsoft.com/en-us/library/cc136992%28VS.85%29.aspx
8. *Understanding* Activation and Submission Filters. Доступ до документів (22.01.2012): http://technet.microsoft.com/en-us/library/ff919469%28WS.10%29.aspx
9. *LINPACK*. Доступ до документів (22.01.2012): http://www.netlib.org/benchmark/hpl
10. *Microsoft* HPC Server 2008 R2. Доступ до документів (22.01.2012): http://www.microsoft.com/hpc/en/us/white-papers.aspx