УДК 004.4'242

І.В. БІЛОКОНЬ[*], С.Д. ПОГОРІЛИЙ[*]

ТЕХНОЛОГІЯ ВІРТУАЛІЗАЦІЇ. ЗАСОБИ ДИНАМІЧНОЇ РЕКОНФІГУРАЦІЇ ОБЧИСЛЮВАЛЬНОГО КЛАСТЕРА

[*]Київський національний університет імені Тараса Шевченка, Київ, Україна

Анотація. *Наведено організаційну структуру динамічно реконфігурованої кластерної обчислювальної системи з вузлами у вигляді віртуальних машин та проаналізовано особливості функціонування її компонент. Запропоновано концепцію побудови автоматичної системи динамічної реконфігурації кластера (АСДРК) і дослідження конфігурацій. Відповідно до сформульованої в роботі концепції запропоновано програмну реалізацію АСДРК.*
Ключові слова: *гнучка архітектура, обчислювальний кластер, відмовостійкий кластер, планувальник завдань, віртуалізація, віртуальна машина, Microsoft Hyper-V, Microsoft Windows HPC Server, динамічна реконфігурація, система автоматичного управління, API.*

Аннотация. *Приведена организационная структура динамически реконфигурируемой кластерной вычислительной системы с узлами в виде виртуальных машин и проанализированы особенности функционирования ее компонент. Предложена концепция построения автоматической системы динамической реконфигурации кластера (АСДРК) и исследования конфигураций. Соответственно сформулированной в работе концепции предложена программная реализация АСДРК.*
Ключевые слова: *гибкая архитектура, вычислительный кластер, отказоустойчивый кластер, планировщик заданий, виртуализация, виртуальная машина, Microsoft Hyper-V, Microsoft Windows HPC Server, динамическая реконфигурация, система автоматического управления, API.*

Abstract. *Organization structure of dynamically reconfigurable computing cluster system with virtual machine nodes is shown and its functional features are analyzed. Automatic dynamic reconfiguration and configuration research system (ADRCRS) creation concept is proposed. According to the proposed concept software implementation of the ADRCRS is suggested.*
Keywords: *flexible architecture, computing cluster, failover cluster, job scheduler, virtualization, virtual machine, Microsoft Hyper-V, Microsoft Windows HPC Server, dynamic reconfiguration, automatic control system, API.*

1. Вступ

У [1, 2] показано актуальність побудови універсальних гнучких обчислювальних систем на базі традиційних архітектур персональних комп'ютерів та серверів (концепція створення гнучких гомогенних архітектур кластерних систем). Згодом цю концепцію було поглиблено за рахунок можливості формування динамічно реконфігурованої кластерної обчислювальної системи з використанням механізмів віртуалізації платформи Microsoft Hyper-V [3, 4]. На основі сформульованої концепції на базі комп'ютерного класу ІТ академії Microsoft Київського національного університету імені Тараса Шевченка було побудовано прототип динамічно реконфігурованої кластерної обчислювальної системи (ДРКОС) з вузлами у вигляді віртуальних машин.

Застосування віртуальних машин (ВМ) в ролі вузлів обчислювального кластера відкриває низку нових можливостей оптимізації використання апаратних обчислювальних ресурсів [3–5]. Оскільки віртуальні машини є певними абстракціями, реконфігурація ДРКОС може бути виконана виключно програмними засобами. В роботі [6] показано вплив конфігурацій ДРКОС на продуктивність роботи програмних реалізацій паралельних алгоритмів. У роботі [5] було сформульовано задачу оптимізації завантаження ресурсів ДРКОС і запропоновано генетичний алгоритм для пошуку оптимальних конфігурацій кластера, а також метод обчислення значень фітнес-функції генетичного алгоритму, що базується на попередньому дослідженні показників завантаженості центрального процесора

для низки так званих «ключових» конфігурацій для кожної обчислювальної задачі і метод виконання експериментального дослідження цих ключових конфігурацій. У роботі [7] запропоновано можливість вдосконалення фітнес-функції шляхом врахування показників завантаженості оперативної пам'яті.

Збір даних для подальшого пошуку оптимальних конфігурацій вимагає створення експериментальної установки, що реалізує сформульовані у роботах [5, 7] методи. Застосуванням описаних у роботах [3, 4] засобів з'являється можливість побудови такої установки у вигляді системи автоматичного управління для автоматичної динамічної реконфігурації і дослідження конфігурацій ДРКОС. Зважаючи на специфіку експлуатації ДРКОС на базі комп'ютерного класу, до автоматичної системи динамічної реконфігурації кластера (АСДРК) і дослідження конфігурацій висуваються такі додаткові вимоги.

1. Можливість автоматичного визначення вільних обчислювальних ресурсів для експерименту.

2. Автоматичний контроль за станом компонентів системи, що приймають участь в експерименті.

3. Запис важливих подій у журнал.

4. Можливість автоматичного запуску при надходженні нових обчислювальних завдань до черги.

У роботі запропоновано концепцію побудови АСДРК, що відповідає сформульованим вимогам. Створені програмні засоби, з одного боку, є інструментом для виконання наукових досліджень, а з іншого – складовою частиною системи автоматичної оптимізації завантаженості обчислювальних ресурсів ДРКОС.

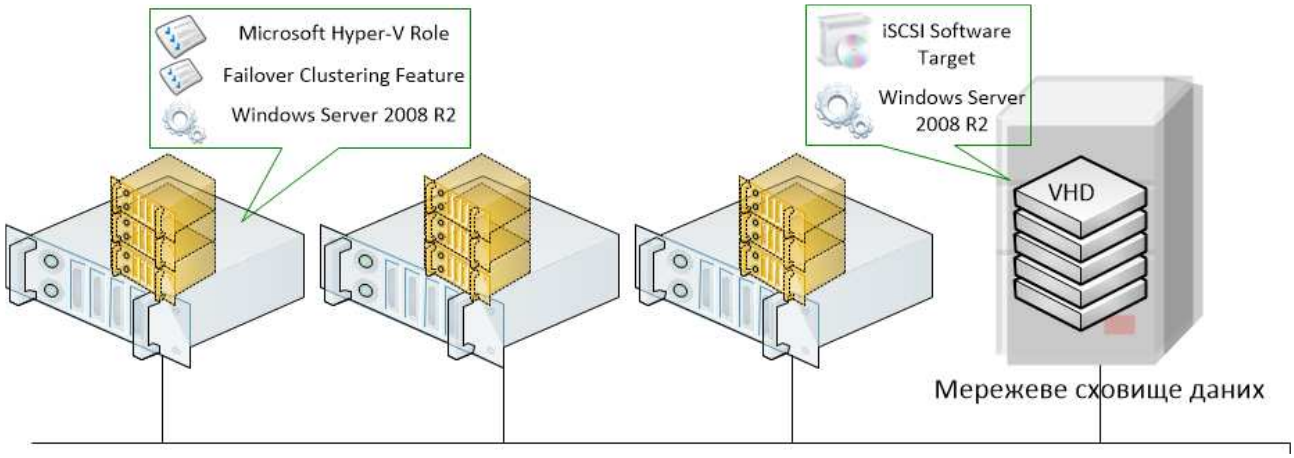
2. Особливості функціонування ДРКОС

Необхідними програмними компонентами для побудови обчислювального кластера на базі персональних комп'ютерів чи серверів є операційна система (ОС) обчислювальних вузлів та диспетчер обчислювальних завдань. Зазвичай для розгортання кластерів використовують програмні пакети, що включають диспетчер завдань, набір прикладних програм для спрощення обслуговування та експлуатації кластера (утиліт), необхідні сервіси, динамічні бібліотеки тощо. У випадку запропонованої ДРКОС додатково до цих програмних засобів необхідна низка компонент для роботи ВМ і реконфігурації обчислювального кластера. Всі ці компоненти є складовими частинами ОС Microsoft Windows Server 2008 R2 Enterprise. Композицію програмно-апаратних компонент ДРКОС розглянуто у роботі [3]. Оскільки ця робота присвячена побудові АСДРК, обмежимося розглядом лише ключових компонент, що використовуються АСДРК.

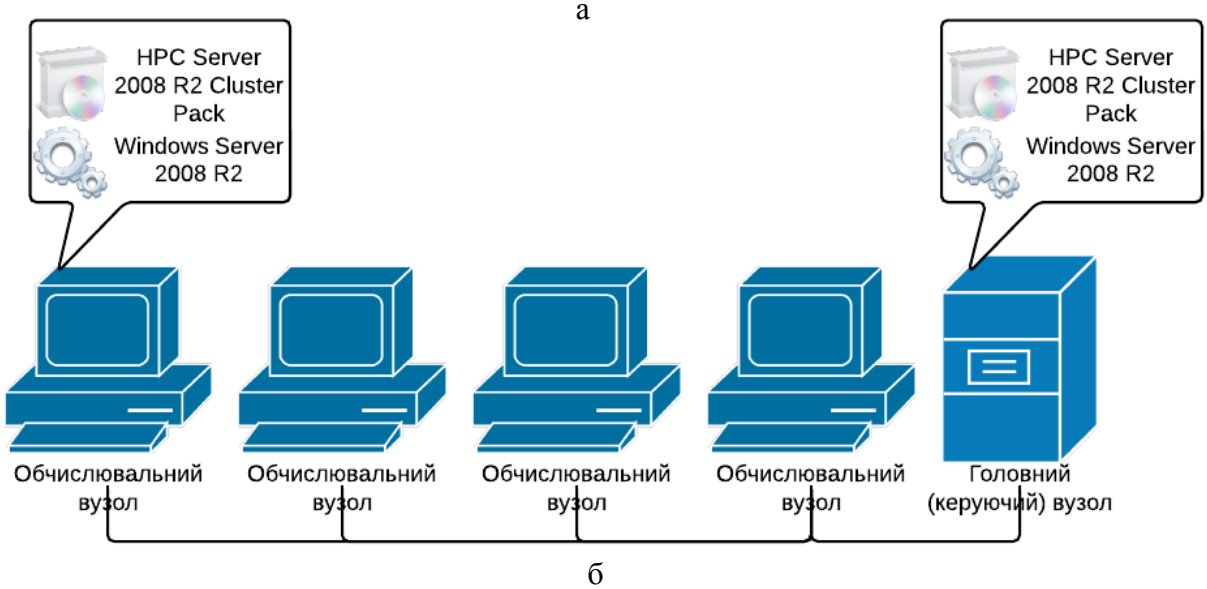
Для ілюстрації взаємозв'язку компонентів ДРКОС розглянемо узагальнене представлення функціональних елементів системи. ДРКОС поєднує в собі три незалежні компоненти, кожна з яких має відповідні прикладні програмні інтерфейси (англ. Application Programming Interface, API) для програмної взаємодії з нею: платформа віртуалізації Microsoft Hyper-V 2008 R2, відмовостійкий кластер (англ. Failover Cluster) на базі Microsoft Windows Server 2008 R2, програмний пакет Microsoft HPC Server 2008 R2. У табл. 1 зведено компоненти ДРКОС, їхній вклад у функціонування системи, а також функції ВМ та серверів ВМ (СВМ) по відношенню до компонентів кластера. Згідно з наведеною таблицею, компоненти ДРКОС формують три незалежні рівні функціонування системи (рівень пакета Microsoft HPC Server (надалі «обчислювальний кластер»), рівень ВМ, рівень відмовостійкого кластера) та відповідно три рівні взаємодії з нею. ВМ та відмовостійкий кластер формують фізичну структуру ДРКОС, а обчислювальний кластер формує відповідно логічну структуру ДРКОС (рис. 1) [3, 4].

Таблиця 1. Компоненти ДРКОС та їх функції у складі системи

Назва компоненти	Основне призначення	Функції, що використовуються в ДРКОС	API	Функція ВМ	Функція СВМ
Платформа віртуалізації Microsoft Hyper-V 2008 R2	Середовище створення і функціонування ВМ	1. Утиліти для адміністрування платформи і ВМ. 2. Зміна станів ВМ. 3. Реконфігурація ВМ	Класи WMI. Простір імен Root\Virtualization [3, 4]	Власне віртуальна машина	Апаратно-програмне середовище для забезпечення функціонування ВМ
Відмовостійкий кластер на базі Microsoft Windows Server 2008 R2	Забезпечення механізмів відмовостійкості для зареєстрованих сервісів	Міграція ВМ	Класи WMI. Простір імен Root\MSCluster [3, 4]	Сервіс відмовості йкого кластера	Вузол відмовостійкого кластера
Програмний пакет Microsoft HPC Server 2008 R2	Набір утиліт та сервісів для створення, адміністрування і експлуатації обчислювального кластера	4. Утиліти для адміністрування і експлуатації обчислювального кластера. 5. Диспетчер завдань: ○ встановлення, налаштування, відміна завдань; ○ моніторинг завдань. 6. Диспетчер вузлів: ○ моніторинг стану вузлів; ○ ініціалізація вузлів; 7. Фільтри [5]	.NET [8]: класи із просторів імен Microsoft.Hpc.Scheduler та Microsoft.Hpc.Scheduler.Properties із бібліотек відповідно Microsoft.Hpc.Scheduler.dll та Microsoft.Hpc.Scheduler.Properties.dll (диспетчер завдань, диспетчер вузлів) [9]. PowerShell [10]: команди оснащення Microsoft.HPC (диспетчер вузлів) [11].	Вузол обчислюва-льного кластера	–



а



б

Рис. 1. Фізична (а) і логічна (б) структури ДРКОС

У процесі розгортання, налаштування на ініціалізації обчислювальні вузли кластера проходять через низку станів, що характеризують ступінь готовності вузла [12].

1. Unknown (невизначений стан) – свідчить про те, що вузол не є частиною обчислювального кластера.

2. Provisioning (підготовка) – виконуються дії щодо приєднання вузла до кластера та його налаштування.

3. Offline (неактивний) – вузол недоступний для запуска на ньому завдань. Натомість над ним можуть бути виконані операції з обслуговування: повторна ініціалізація, від'єднання від кластера, налаштування тощо.

4. Starting (запуск) – свідчить про виконання дій щодо переходу вузла до активного стану.

5. Online (активний) – вузол готовий до запуску на ньому завдань кластера. Коли вузол перебуває у цьому стані, не можна виконувати операції з його обслуговування.

6. Removing (вилучення) – виконуються операції щодо від'єднання вузла від кластера.

У результаті аналізу цих станів та переходів між ними було сформовано діаграму, зображену на рис. 2. Згідно з наведеною діаграмою, переходи зі станів Unknown, Offline, Online відбуваються внаслідок виконання відповідних операцій над вузлами, а переходи зі станів Provisioning, Starting, Removing відбуваються автоматично. Таким чином, стани Provisioning, Starting та Removing є проміжними.

Нижче розглянуто операції переходів більш детально.

1. Assign Node Template (призначити шаблон вузла) – приєднує вузол до обчислювального кластера (або виконує повторну ініціалізацію), виконує необхідні для цього налаштування і реєструє характеристики доступних на вузлі обчислювальних ресурсів.

2. Bring online (активувати) – переводить вузол у стан Online.

3. Take offline (деактивувати) – переводить вузол у стан Offline.

4. Delete (вилучити) – від'єднує вузол від кластера.

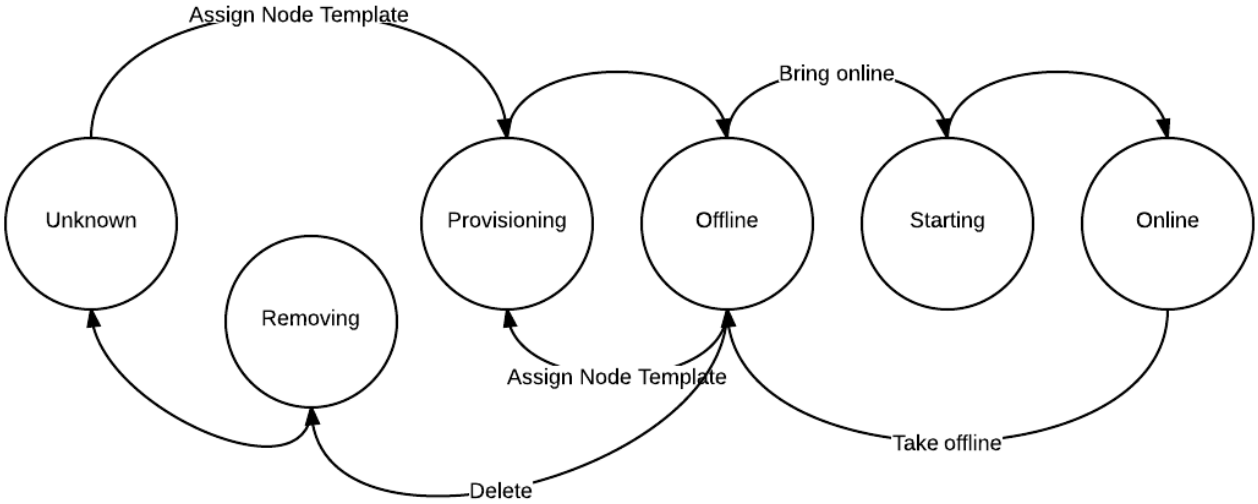


Рис. 2. Стани готовності вузлів обчислювального кластера та операції переходів

Всі операції з обчислювальними вузлами (контроль стану, налаштування параметрів, встановлення/моніторинг завдань, запуск процесів програм, створених з використанням Message Passing Interface (MPI) тощо) здійснюються за рахунок відповідних сервісів ОС Windows, що встановлюються при розгортанні пакета HPC Server 2008 R2 на вузлах. Список сервісів обчислювального кластера та їх призначення наведено у табл. 2. Сервіс «HPC Job Scheduler Service» головного вузла періодично надсилає запити до сервісу «HPC Node Manager Service» кожного зареєстрованого обчислювального вузла для перевірки його доступності. Якщо перевірка успішна, то стан доступності приймає значення «OK», якщо ні – «Unreachable» (недоступний). Таким чином, для можливості запуску завдань на вузлі необхідно, щоб його стан готовності був Online, а стан доступності – OK.

Обчислювальні ресурси кластера виділяються диспетчером завдань. Завдання (англ. job) фактично є запитом на виділення ресурсів, визначає обмеження на кількість та якість обчислювальних ресурсів, параметри доступу і може містити одну і більше задач (англ. task). Задача визначає налаштування для запуску виконуваного файлу чи команди (параметри командного рядка, вхідні і вихідні дані, необхідну кількість обчислювальних ресурсів для виконання). Налаштування параметрів завдання і його задач зберігається у файлі формату XML.

Таблиця 2. Сервіси обчислювального кластера

Назва	Зареєстроване в ОС ім'я	Функція	Головний вузол	Обчислювальний вузол
HPC Job Scheduler Service	HpcScheduler	Диспетчеризація, встановлення, налаштування, моніторинг, відміна завдань обчислювального кластера, перевірка стану вузлів, перевірка прав доступу до обчислювальних ресурсів	+	□
HPC	HpcManagement	Управління налаштування-	+	+

Продовж. табл. 2

Management Service	HpcManagement	ми кластера і його вузлів за допомогою API	+	+
HPC Node Manager Service	HpcNodeManager	Керування процесами завдань кластера на вузлах, звітування про стан вузлів	+	+
HPC MPI Service	msmpi	Запуск MPI процесів	+	+

Для збільшення гнучкості налаштування диспетчеризації завдань застосовують вбудований обчислювальний кластер: механізм фільтрів. Диспетчер завдань, у випадку наявності фільтрів, використовує їх для прийняття остаточного рішення про подальшу долю завдання. Фільтр являє собою застосування, на вхід якого подається шлях до файлу налаштувань завдання. В залежності від значення, що повертається, диспетчером приймається відповідне рішення. Більш детально цей механізм описано у роботі [5].

Обчислювальний кластер може розпочати виконання завдання лише у випадку наявності вільних ресурсів такого типу та в такій кількості, яка зазначена у налаштуваннях завдання. Тип ресурсу визначає квант обчислювального ресурсу, що буде виділено кожному процесу завдання. Параметр типу ресурсу може приймати значення «вузол» («node»), «ядро» («core») і «сокет» («socket» – процесор у багатопроцесорних вузлах).

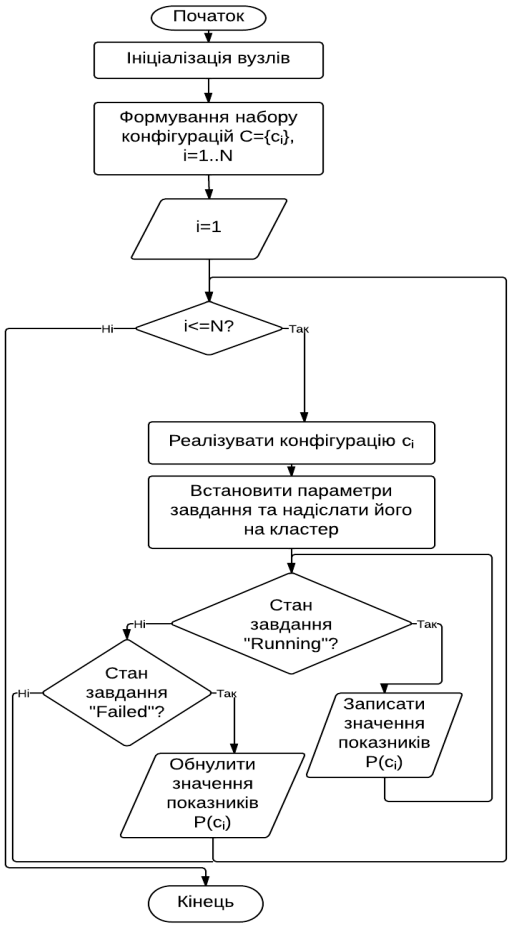


Рис. 3. Блок-схема алгоритму дослідження конфігурацій

3. Методи дослідження конфігурацій ДРКОС

Метою дослідження конфігурацій ДРКОС є отримання для обчислювальної задачі даних про певні показники продуктивності для кожної ключової конфігурації [5] від ОС кожного вузла, задіяного під цю задачу. На рис. 3 наведено узагальнену принципову блок-схему алгоритму дослідження. Оскільки зазвичай обчислювальні завдання працюють тривалий час, експеримент варто виконувати протягом деякого невеликого попередньо зазначеного проміжку часу, після чого примусово завершувати виконання поточного завдання і переходити до наступної конфігурації.

Під час отримання показників необхідно враховувати стан завдання. Так, якщо протягом експерименту завдання завершилося невдачею (стан «Failed»), неможливо об’єктивно однозначно обчислити значення показників через неможливість однозначного програмного визначення причини помилки. Тому у такому випадку будемо вважати, що досліджувана конфігурація є непридатною для поточного завдання і забезпечуємо відображення цього факту у вихідних даних (покладаємо значення показників рівними нулю).

Процес дослідження конфігурацій ДРКОС являє собою композицію синхронних та асинхронних операцій, що включають взаємодію із різними компонентами системи. Розглянемо ці операції більш детально.

Ініціалізація вузлів

Для виконання досліджень конфігурацій ДРКОС необхідне попереднє визначення обчислювальних ресурсів, на яких виконуватиметься експеримент. Побудова ДРКОС на базі комп'ютерного класу передбачає можливість використання певної частини апаратних засобів для цілей, не пов'язаних із обчислювальним кластером, що разом із наявністю активних завдань на кластері є причиною неможливості статичного виділення ресурсів для досліджень. Тому виникає необхідність динамічного визначення доступних для експерименту ВМ та СВМ, підпорядкованих ДРКОС.

Рекомендації щодо застосування віртуалізації серверів для робочих навантажень вимагають використання СВМ виключно як апаратне середовище для ВМ. Тобто будь-які робочі навантаження повинні працювати лише у ВМ. При визначенні вільних СВМ будемо вважати, що ці рекомендації виконано (інакше в загальному випадку неможливо визначити наявність навантажень на СВМ). Таким чином, однією із умов можливості використання апаратних ресурсів СВМ для експерименту є відсутність активних ВМ, що не є вузлами ДРКОС. Другою умовою є відсутність на СВМ вузлів ДРКОС із активними завданнями.

Усі ВМ, що є вузлами ДРКОС, не містять активних завдань і знаходяться у стані доступності «OK», будемо вважати доступними для дослідження. Для реалізації всіх ключових конфігурацій достатньо, щоб кількість доступних вузлів була рівною кількості доступних СВМ. У випадку, коли кількість доступних вузлів більша кількості доступних СВМ, зайві вузли необхідно вимкнути для уникнення можливого небажаного впливу на результати експериментів. Якщо кількість доступних вузлів менша кількості доступних СВМ і є вимкнені вузли, необхідно увімкнути потрібну кількість вузлів.

Формування конфігурацій ДРКОС

Для можливості програмного перебору конфігурацій необхідно визначити метод їх кодування, а для їх реалізації і дослідження, відповідно, метод декодування у термінах операцій над компонентами ДРКОС і їх параметрів. Параметри операцій визначаються відповідними функціями API. Реконфігурація ресурсів ВМ здійснюється методом ModifyVirtualSystemResources класу Msvm_VirtualSystemManagementService [13], а переміщення ВМ – методом MoveToNewNode класу MSCluster_ResourceGroup (швидка міграція) або методом ExecuteResourceControl класу MSCluster_Resource (жива міграція) [14].

У роботі [4] наведено список із 9 основних операцій, що можуть бути виконані над вузлами ДРКОС для динамічної реконфігурації ресурсів. У роботі [5] запропоновано представлення досліджуваних конфігурацій ДРКОС у вигляді конфігурацій розташування процесів завдання відносно ВМ. Оскільки АСДРК повинна реалізовувати методи дослідження із [5], при формуванні конфігурацій скористаємося відповідними міркуваннями і наближеннями:

- усі процеси досліджуваного завдання алгоритмічно ідентичні і не мають внутрішнього розпаралелювання на рівні потоків;
- кількість процесів завдання парна;
- усі СВМ, підпорядковані ДРКОС, ідентичні;
- з міркувань забезпечення універсальності методу як параметри конфігурацій доцільно використовувати лише кількісні та якісні характеристики обчислювальних ресурсів: центрального процесора (ЦП), оперативної пам'яті (ОП) та мережевого з'єднання.

Методи реконфігурації ресурсів ЦП, ОП, мережевого з'єднання ВМ та набір їх параметрів відповідно до API наведено у табл. 3.

Таблиця 3. Методи реконфігурації ресурсів ВМ та їх параметри

№	Назва	Параметри		
		Назва параметра	Тип	Опис
1	Реконфігурація ресурсів ОП	DynamicMemoryEnabled	boolean	Увімкнено режим динамічної ОП
		Limit	uint64	Максимальний об'єм ОП, що може бути спожитий ВМ
		Reservation	uint64	Об'єм ОП, що гарантовано має бути виділений ВМ
		TargetMemoryBuffer	uint32	Визначає частку виділеної ОП, що буде зарезервовано додатково до виділеної ОП
		VirtualQuantity	uint64	Об'єм ОП, необхідний для запуску ВМ
		Weight	uint32	Пріоритет виділення ресурсів динамічної ОП
2	Реконфігурація ресурсів ЦП	VirtualQuantity	uint64	Кількість ядер ВМ
		Weight	uint32	Пріоритет виділення ресурсів ЦП
3	Швидка міграція ВМ	NodeName	string	Ім'я або IP-адреса цільового СВМ
		TimeOut	uint32	Проміжок часу, виділений на операцію
4	Жива міграція ВМ	ControlCode	sint32	Код операції (приймає значення 23068676)
		InputBuffer	uint8 []	Ім'я або IP-адреса цільового СВМ у вигляді масиву байтів кодування UTF-8

Зважаючи на ідентичність параметрів та сталість апаратної конфігурації мережевого з'єднання між СВМ, для зменшення кількості параметрів конфігурацій при їх кодуванні мережеве з'єднання доцільно позначати як взаємне розташування ВМ (відносні координати ВМ). Симетрія відносно апаратних характеристик СВМ та мережі, а також наведені вище наближення дають можливість переходу від конфігурацій ЦП та відносних координат ВМ до конфігурацій розташування процесів завдань відносно ВМ при кодуванні досліджуваних конфігурацій. Таким чином, для реконфігурації в межах запропонованого дослідження доцільно застосовувати операції над ВМ, наведені у табл. 4. Згідно з таблицею, будь-яку досліджувану конфігурацію можна представити як набір параметрів операцій (1)–(5) у вигляді пар («ключ», «значення»), де ключем є унікальний ідентифікатор ВМ, а значенням – цільове значення параметра операції.

Таблиця 4. Відповідність параметрів операцій із реконфігурації ВМ параметрам методів API

№	Операція	Параметри ([Значення] – цільове значення параметра операції)	
		Назва	Значення
1	Модифікація об'єму ОП	DynamicMemoryEnabled	false
		Limit	[Значення]
		Reservation	[Значення]
		VirtualQuantity	[Значення]
2	Модифікація кількості ЦП	VirtualQuantity	[Значення]
3	Модифікація типу ОП	DynamicMemoryEnabled	[Значення]
4	Автоматична зміна	DynamicMemoryEnabled	true

Продовж. табл. 4

	об’єму ОП		
5	Швидка міграція	NodeName	[Значення]
		TimeOut	uint32.MaxValue

Для зручності перебору конфігурацій розташування процесів скористаємось методом їх кодування, запропонованим у роботі [7]. Будемо позначати кожну конфігурацію трьома числами: r (загальна кількість процесів завдання), p (максимальна кількість процесів на ВМ), g (максимальна кількість ВМ із процесами завдання на СВМ). Причому, при декодуванні конфігурацій і представленні їх у вигляді параметрів операцій над елементами ДРКОС необхідно забезпечувати максимальну відповідність значенням параметрів конфігурації. Таким чином, конфігурації, що прийматимуть участь у дослідженні (ключові), мають такі значення параметрів [7]:

- К1: $r,1,1$;
- К2: $\begin{cases}(r, C_{\max}, r \div C_{\max}), r > C_{\max} \vee r \vdots C_{\max} = 0 \\ (r, C_{\max}, r \div C_{\max} + 1), r > C_{\max} \vee r \vdots C_{\max} > 0 \\ (r, r, 2), r < C_{\max}\end{cases}$;
- К3: $\begin{cases}(r, C_{\max}, 1), r > C_{\max} \\ (r, r, 1), r < C_{\max}\end{cases}$.

Тут $C_{\max}$ − кількість паралельних ядер ЦП СВМ.

Для виконання реконфігурації ДРКОС значення r, p, g кожної конфігурації необхідно декодувати у фактичні значення параметрів операцій (2) та (5) (табл. 4). Ці операції та їх параметри у даному випадку досить тісно пов’язані. Наприклад, конфігурація (6, 4, 1) відповідає розміщенню на різних СВМ 2 ВМ із 4 та 2 ЦП відповідно. Для реалізації такої конфігурації необхідно по-перше, розмістити відповідним чином ВМ, а по-друге, виконати відповідну модифікацію кількості ЦП на цих ВМ. Декодування можна здійснити багатьма способами. В цій роботі пропонується виконувати декодування з міркувань незалежності від порядку вибору конфігурацій, мінімізації кількості ВМ, що представляють конфігурацію та мінімізації часу виконання реконфігурації. Найтривалішою операцією в даному випадку є операція (2), оскільки вона потребує перезапуску ВМ. Тому в алгоритмі декодування передбачено вибір цільових ВМ із списку доступних таким чином, щоб при можливості уникнути реконфігурації ЦП ВМ. Алгоритм наведено нижче.

1. Формуємо поточну конфігурацію розташування ВМ у вигляді множини $\{P_1, P_2, ..., P_i\}$, де $P_i \subset V, i = 1,2,..., N$ (V – множина доступних ВМ, P_i – множина віртуальних машин на СВМ з індексом *i*), упорядкованих за спаданням $|P_i|$.

2. $i = 1$.

3. $i > N$? Так – перехід до п. 19.

4. $|P_i| = g$? Так – перехід до п. 18.

5. $|P_i| < g$? Так – перехід до п. 11.

6. $j = 1$.

7. $j > N$? Так – перехід до п. 18.

8. $|P_j| \geq g$? Так – перехід до п. 10.

9. $P_i = P_i \setminus \{p_{i1}\}, P_j = P_j \cap \{p_{i1}\}, p_{i1} \in P_i$ – перший елемент множини. Перехід до п. 4.

10. $j = j + 1$. Перехід до п. 7.

11. $\forall_{P_k} |P_k| < g : \sum_k |P_k| < g$? Так – перехід до п. 19.

12. $j = 1$.

13. $j > N$? Так – перехід до п. 18.

14. $(P_j = P_i) \vee (|P_j| = g) \vee (|P_j| = 0)$? Так – перехід до п. 17.

15. $P_i = P_i \cap \{p_{j1}\}, P_j = P_j \setminus \{p_{j1}\}, p_{j1} \in P_j$.

16. $|P_i| < g$? Так – перехід до п. 12.

17. $j = j + 1$. Перехід до п. 13.

18. $i = i + 1$. Перехід до п. 3.

19. $\Pi = \{B_k \mid \forall_{P_k} |P_k| = g : B_k = P_k\}$ – множина наборів ВМ, розташування яких точно відповідає конфігурації. $Y = \phi$ – множина ВМ, що складають конфігурацію.

20. $k = 1$.

21. $k > |\Pi|$? Так – перехід до п. 31.

22. $d = r$.

23. $j = 1$.

24. $j > |B_k|$? Так – перехід до п. 30.

25. $d \geq p$? Так: $s = p$. $d = d - p$. Перехід до п. 27.

26. $s = d$. $d = 0$.

27. Встановити кількість ЦП для $v_j \in B_k$ рівну s. $Y = Y \cap \{v_j\}$.

28. $d \leq 0$? Так – перехід до п. 31.

29. $j = j + 1$. Перехід до п. 24.

30. $k = k + 1$. Перехід до п. 21.

31. Завершення.

Цей алгоритм надає переваги при послідовному декодуванні конфігурацій К2 і К3, якщо після першої його ітерації множину $\{P_i\}$ ініціалізувати множиною $\{B_k\}$. Тоді зміна розташування ВМ не призведе до необхідності внесення змін у конфігурацію ЦП, оскільки К2 і К3 відрізняються лише параметром g.

Дослідження конфігурацій

Процес дослідження конфігурації ДРКОС включає такі етапи:

- реалізація конфігурації;
- встановлення параметрів завдання і надсилання його на кластер;
- збір даних про показники продуктивності.

Розглянемо ці етапи більш детально.

Згідно з [4], частина операцій над ВМ передбачає їх перезавантаження та вимагає сповіщення обчислювального кластера про зміну кількості обчислювальних ресурсів відповідних вузлів. Розглянемо алгоритм реалізації конфігурації, що вимагає перезавантаження ОС ВМ.

1. Переводимо вузли, що підлягають реконфігурації, у стан «Offline».
2. Вимикаємо відповідні вузли.
3. Виконуємо реконфігурацію ресурсів.
4. Вмикаємо відповідні вузлі.
5. У випадку динамічного призначення IP-адрес вузлів за допомогою Dynamic Host Configuration Protocol (DHCP), виконуємо очищення кеша DNS головного вузла.

6. Очікуємо готовності вузлів. Вузол вважається готовим, якщо стан доступності вузла – «OK» та на ньому запущено всі сервіси, наведені у табл. 2.

7. Виконуємо повторне призначення шаблонів вузлам.

8. Переводимо вузли у стан «Online».

Після цього вузли готові до запуску завдань. Тепер необхідно встановити параметри завдання відповідно до сформованої конфігурації.

API обчислювального кластера надає можливість встановлення значень цілої низки властивостей завдань (клас SchedulerJob) і задач (SchedulerTask) перед надсиланням їх на кластер. У табл. 5 наведено список властивостей та їх значення, що застосовуються при дослідженні конфігурації.

Таблиця 5. Властивості класу SchedulerJob та їх значення при дослідженні конфігурації

Властивість	Значення	Опис
MinimumNumberOfCores	r	Мінімальна кількість доступних ядер
MaximumNumberOfCores	r	Максимальна кількість доступних ядер
Runtime	Час дослідження конфігурації (сек)	Час виконання завдання
UnitType	«Core»	Тип ресурсу для завдання
RequestedNodes	Y	Імена вузлів, на які буде встановлено завдання

Клас SchedulerTask також має властивості MinimumNumberOfCores та MaximumNumberOfCores, сума відповідних значень яких для всіх задач у складі завдання не повинна виходити за рамки, встановлені на рівні завдання. Оскільки в роботі розглянуто найбільш розповсюджений випадок (завдання має одну задачу, що повинна виконуватись на кожному вузлі), ці властивості також повинні приймати значення r. Таким чином ми повідомляємо обчислювальний кластер про те, що завдання повинне виконуватися виключно на r ядрах по одному екземплярі завдання на кожному ядрі.

Протягом дослідження конфігурації запропонована АСДРК збирає дані лічильників продуктивності кожного вузла, що виконує завдання, відповідно до експерименту. Кожен лічильник ОС Windows визначається трьома параметрами: CategoryName (назва категорії – об'єкт дослідження), CounterName (назва лічильника), InstanceName (назва екземпляра у випадку наявності декількох екземплярів). Таким чином, згідно з запропонованим методом дослідження, АСДРК отримує інформацію із таких лічильників: завантаженість ЦП (CategoryName=«Processor», CounterName=«% Processor Time», InstanceName=«_Total») та виділена ОП (CategoryName=«Memory», CounterName=«Committed Bytes», InstanceName=««). Оскільки фактичні значення показників потребують попередньої обробки для подальшого їх застосування, АСДРК виконує відповідну обробку протягом експерименту. Так, показники завантаженості ЦП у вихідних результатах представлено як усереднений за часом і по вузлах інтеграл функції завантаженості ЦП [5], а показники ОП – максимальне значення виділеної пам'яті, усереднене по вузлах.

Програмна реалізація

Програмна реалізація запропонованої АСДРК виконана у вигляді застосування мовою C# (.NET 3.5), розгорнутого на головному вузлі. Вхідним параметром застосування є шлях до XML-файлу завдання. Додаткові параметри (список СВМ, час проведення дослідження конфігурації тощо) програма отримує із конфігураційного файлу. АСДРК може працювати у 2 режимах: у режимі фільтра передачі [5] та звичайного консольного застосування.

Вхідний XML-файл аналізується і, у випадку виявлення недопустимих для експериментального дослідження значень параметрів, генеруються відповідні виключення. На основі отриманих даних формується діапазон досліджуваних конфігурацій відносно параметра r: дослідження виконуються для парних значень r від $r = \text{MinimumNumberOfCores}$ до $r = \text{MaximumNumberOfCores}$. Відповідно до поточного значення r формуються значення параметрів p і g.

Результати дослідження виводяться у текстові файли у форматі:

r p g значення1 значення2 …

При такому форматі у вихідний файл можна виводити результати дослідження довільної кількості лічильників продуктивності. Приклад вихідного файлу наведено нижче.

2 1 1 35.8120593831386 1201405952

2 2 1 74.916674214753 642568192

Імена вихідних файлів являють собою значення хеш-функцій від командного рядка у налаштуванні задачі. Таке іменування забезпечує розмежування отриманих даних для різних виконуваних файлів та їх параметрів.

У процесі роботи АСДРК важливо фіксувати події (виключні ситуації, етапи експерименту тощо) у журналі для можливості відлагодження чи спостереження за ходом експерименту. Кожен запис у журналі подій запропонованої АСДРК містить такі дані: час виникнення події, тип події (помилка або повідомлення), повідомлення (додаткова інформація про подію), джерело (сутності або параметри операції, що призвели до виникнення повідомлення), операція (що призвела до виникнення повідомлення).

4. Висновки

1. Відповідно до сформульованого методу запропоновано концепцію побудови АСДРК для автоматичного дослідження конфігурацій ДРКОС з вузлами у вигляді ВМ, яка дозволяє отримати АСДРК із такими властивостями:

• можливість автоматичного визначення вільних обчислювальних ресурсів для експерименту;

• автоматичний контроль за станом компонентів системи, що приймають участь в експерименті;

• запис важливих подій у журнал;

• можливість автоматичного запуску при надходженні нових обчислювальних завдань до черги.

2. У результаті виконаного аналізу принципів функціонування компонентів ДРКОС запропоновано алгоритми реалізації та дослідження ключових конфігурацій. Визначені операції для реконфігурації ДРКОС та їх параметри, а також співвідношення між параметрами операцій та функцій API компонентів ДРКОС дають можливість програмної реалізації АСДРК для дослідження ДРКОС на платформі Microsoft Windows HPC Server із вузлами у вигляді ВМ на платформі Microsoft Hyper-V.

3. Відповідно до запропонованої концепції виконано програмну реалізацію АСДРК, яка перебуває у стадії дослідної експлуатації і дає досліднику можливість автоматичного отримання даних про середню завантаженість ЦП і потреби в ОП вузлів ДРКОС для оцінки якості конфігурацій і подальшого формування, модифікації і дослідження фітнес-функції.

СПИСОК ЛІТЕРАТУРИ

1. Концепція створення гнучких гомогенних архітектур кластерних систем / С.Д. Погорілий, Ю.В. Бойко, Д.Б. Грязнов [та ін.] // Матеріали шостої міжнар. наук.-практ. конф. з програмування

УкрПРОГ`2008. Проблеми програмування, (Київ, 27 – 29 травня 2008 р.). − 2008. − № 2−3. − С. 84 – 90.
2. Білоконь І. Створення захищеного обчислювального windows-кластеру на платформі desktop PC / І. Білоконь, Д. Грязнов, В. Мар'яновський // Вісник Київського національного університету імені Тараса Шевченка. – (Серія «Радіофізика та електроніка»). – К., 2009. – Вип. 12. – С. 13 – 16.
3. Білоконь І. Побудова динамічно реконфігурованої обчислювальної архітектури з використанням технологій віртуалізації / І. Білоконь, Д. Грязнов, С. Погорілий // Вісник Київського національного університету імені Тараса Шевченка. – (Серія «Радіофізика та електроніка). – К., 2010. – Вип. 14. – С. 4 – 6.
4. Погорілий С. Технологія віртуалізації. Динамічна реконфігурація ресурсів обчислювального кластера / С. Погорілий, І. Білоконь, Ю. Бойко // Математичні машини і системи. – 2012. – № 3. – С. 3 – 18.
5. Погорілий С.Д. До задачі оптимізації завантаженості ресурсів обчислювального кластера з вузлами у вигляді віртуальних машин / С.Д. Погорілий, І.В. Білоконь // Матеріали 8 міжнар. наук.-практ. конф. з програмування «УкрПРОГ–2012». Проблеми програмування, (Київ, 22–24 травня 2012 р.). – Київ, 2012. – № 2–3. – С. 93 – 101.
6. Білоконь І. Дослідження впливу конфігурації обчислювальної системи на продуктивність програмних реалізацій паралельних алгоритмів / І. Білоконь, Д. Грязнов, С. Погорілий // Праці VII Міжнар. конф. «Електроніка та прикладна фізика», (Київ, 19–22 жовтня 2011 р.). – Київ, 2011. – С. 181– 182.
7. Bilokon I. Research of Genetic Algorithm for searching optimal configurations of computing cluster with virtual machine nodes / I. Bilokon, S. Pogorilyy // Theoretical and Applied Aspects of cybernetics. Proc. of the 2nd International Scientific Conference of students and Young Scientists. – Kyiv: Bukrek, 2012. – P. 13 – 18.
8. NET Framework [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com/en-us/vstudio/aa496123.aspx.
9. Microsoft High Performance Computing for Developers [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com/en-us/library/ff976568.aspx.
10. Windows PowerShell [Електронний ресурс]. – Режим доступу: http://technet.microsoft.com/en-us/library/bb978526.aspx.
11. Windows HPC Cmdlets for Windows PowerShell [Електронний ресурс]. – Режим доступу: http://technet.microsoft.com/en-us/library/ff950195.aspx.
12. Understanding Node States, Health and Operations [Електронний ресурс]. – Режим доступу: http://technet.microsoft.com/en-us/library/cc972847(v=ws.10).aspx#BKMK_Health.
13. Hyper-V WMI Classes [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com/en-us/library/cc136986(v=vs.85).aspx.
14. Failover Cluster Provider Reference [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com/en-us/library/windows/desktop/aa372876(v=vs.85).aspx.

Стаття надійшла до редакції 20.05.2013