УДК 004.4'242

С.Д. ПОГОРІЛИЙ, І.В. БІЛОКОНЬ, Ю.В. БОЙКО

ТЕХНОЛОГІЯ ВІРТУАЛІЗАЦІЇ. ДИНАМІЧНА РЕКОНФІГУРАЦІЯ РЕСУРСІВ ОБЧИСЛЮВАЛЬНОГО КЛАСТЕРА

Анотація. *На основі виконаного аналізу сучасних платформ апаратно-програмної віртуалізації обґрунтовано доцільність використання і дослідження віртуальних машин на платформі Microsoft Hyper-V R2 як вузлів обчислювального кластера. Концепцію створення гнучких гомогенних архітектур кластерних систем поглиблено за рахунок можливості формування динамічно реконфігурованої кластерної обчислювальної системи з використанням механізмів віртуалізації платформи Microsoft Hyper-V. Показано актуальність використання апаратної платформи персональних комп'ютерів і серверів для реконфігурації ресурсів обчислювальних кластерів.*
Ключові слова: *гнучка архітектура, обчислювальний кластер, планувальник завдань, віртуалізація, віртуальна машина, Microsoft Hyper-V, Microsoft HPC Server, хмарні обчислення, оптимізація, продуктивність обчислень.*

Аннотация. *На основании выполненного анализа современных платформ аппаратно-программной виртуализации обоснована целесообразность использования и исследования виртуальных машин на платформе Microsoft Hyper-V R2 в качестве узлов вычислительного кластера. Концепция создания гибких гомогенных архитектур кластерных систем углублена за счет возможности формирования динамически реконфигурируемой кластерной вычислительной системы с использованием механизмов виртуализации платформы Microsoft Hyper-V. Показана актуальность использования аппаратной платформы персональных компьютеров и серверов для реконфигурации ресурсов вычислительных кластеров.*
Ключевые слова: *гибкая архитектура, вычислительный кластер, планировщик заданий, виртуализация, виртуальная машина, Microsoft Hyper-V, Microsoft HPC Server, облачные вычисления, оптимизация, производительность вычислений.*

Abstract. *On the basis of performed analysis of modern hardware and software virtualization platforms it is proved the practicability of usage and researching of virtual machines on the Microsoft Hyper-V R2 platform as a compute cluster nodes. The concept of flexible homogeneous cluster architectures construction was expanded with ability of dynamically reconfigurable cluster computing system implementation using Microsoft Hyper-V technology virtualization features. The urgency of hardware platform of personal computers and servers for reconfiguration of the resources of computing clusters is shown.*
Keywords: *flexible architecture, computing cluster, task scheduler, virtualization, virtual machine, Microsoft Hyper-V, Microsoft HPC Server, cloud computing, optimization, computational performance.*

1. Вступ

Програмні реалізації паралельних алгоритмів мають різний ступінь реалізованого паралелізму та висувають широкий спектр вимог до міжвузлового, між'ядерного, міжпроцесного та міжпотокового зв'язку. Тому актуальною проблемою є побудова колективного обчислювального ресурсу, який задовольняє потребам широкого кола обчислювальних задач і належить до систем, що динамічно реконфігуруються. Традиційним підходом до їх створення є реалізація проблемно-орієнтованих обчислювальних систем на базі програмованих логічних інтегральних схем (ПЛІС) [1–7]. У даній галузі велика кількість результатів представлена в роботах Палагіна О.В.,

ISSN 1028-9763. Математичні машини і системи, 2012, № 3

Опанасенка В.Н., Кургаєва О.П., Каляєва І.А. Обмеженнями таких обчислювальних систем є відсутність універсальності та необхідність у спеціалізованих засобах розробки програмного забезпечення (ПЗ).

Величезна кількість ПЗ і модулів, реалізованих 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 в залежності від потреб, часу доби тощо [8–11]. Традиційним шляхом розподілу обчислювальних ресурсів кластера є застосування планувальників завдань. Планувальник (або диспетчер) – це застосування, що виконує запуск обчислювальних завдань на кластері відповідно до певного порядку. Цей порядок визначається зазвичай метаданими задачі, що вказуються користувачем під час її реєстрації. Часто алгоритм планувальника приймає рішення про надання обчислювальних ресурсів задачам, додатково базуючись на даних про стан вузлів, що отримуються в процесі експлуатації кластера. В будь-якому випадку ресурси надаються певній задачі до її запуску. Такий підхід пов'язаний із низкою недоліків, головними з яких є відсутність врахування логіки роботи задачі і динамічної зміни в ресурсах для задачі, не враховується зміна стану вузлів у процесі її роботи (наприклад, задача може на певному етапі потребувати більше оперативної пам'яті (ОП) і процесорного часу, ніж їй було надано, або навпаки – може не використовувати всіх наданих їй ресурсів, що призводить до їх неефективного використання).

Сучасні досягнення *у* технологіях апаратно-програмної віртуалізації дали можливість одночасної роботи на одному комп'ютері декількох примірників незалежних операційних систем (ОС) з мінімальним фактором взаємовпливу. Продуктивність таких віртуальних серверів майже не поступається продуктивності фізичних із аналогічною конфігурацією, але в цьому випадку віртуалізація дозволяє використовувати апаратні можливості комп'ютера більш раціонально та змінювати конфігурацію обчислювальних ресурсів віртуального сервера без внесення змін до апаратного забезпечення (АЗ) фізичної платформи. Таким чином, віртуальні машини (ВМ) стають новими кандидатами на платформу для створення обчислювальних кластерів, перетворюючи ПК і сервери із фактичних обчислювальних вузлів на середовище існування більш гнучких обчислювальних архітектур.

2. Побудова обчислювальних кластерів на базі ПК і серверів

Обчислювальні кластери почали свій шлях на початку 70-х років як можливість нарощування обчислювальної продуктивності суперкомп'ютерів. Історія їх розвитку нерозривно пов'язана із розвитком комп'ютерних мереж. Так, у 1971 році було побудовано обчислювальний кластер C.mpp [12], а з появою механізмів розподілу задач і файлів по мережі у 1983 р. почали з'являтися кластерні ОС та відповідні інструментарії паралельного програмування.

Історія створення обчислювальних кластерів на базі ПК та серверів багато в чому завдячує проекту Parallel Virtual Machine (PVM) [13]. Створення кластерів на основі дешевих персональних комп'ютерів було продовжено Американським аерокосмічним агентством (NASA), а згодом, у 1995 році, з'явилися кластери Beowulf [10] (обчислювальні кластери, побудовані на широкодоступних апаратних засобах та ПЗ з відкритими вихідними кодами).

Розвиток мережі Internet, концепції хмарних обчислень [14–16] та апаратно-програмної віртуалізації дав можливість створення обчислювальних кластерів, вузли якого є ВМ, що працюють у «хмарі» (сервіси HPC Azure корпорації Microsoft та EC2 компанії

Amazon, приватні «хмари»). Тенденція збільшення популярності таких сервісів свідчить про подальші перспективи використання організаціями орендованих обчислювальних ресурсів замість розгортання і підтримки власних потужностей. Та, незважаючи на це, побудова власних обчислювальних кластерів залишається актуальною принаймні для навчальних цілей, обрахунків даних, що становлять таємницю, створення надпотужних обчислювальних систем тощо.

Апаратні платформи

Більшість сучасних обчислювальних кластерів, що входять до Top 500 найпотужніших у світі [17], побудовано на базі багатопроцесорних серверів та спеціалізованих обчислювальних модулів, об'єднаних високошвидкісним мережевим зв'язком (Infiniband, 10 Gigabit Ethernet, FiberChannel, Myrinet). Доступною альтернативою таким системам є обчислювальні кластери, побудовані на базі парку традиційних ПК, об'єднаних мережею Gigabit Ethernet. Обчислювальну потужність ПК для деяких задач можна значно збільшити за рахунок використання сучасних відеоадаптерів для виконання розрахунків загального призначення (технології GPGPU – General-purpose graphics processing units) [18, 19]. Таким чином, до актуальності створення подібних обчислювальних кластерів, наведеної у [8], варто додати можливість використання відеоадаптерів для виконання високопродуктивних обчислень (технології NVidia Cuda, ATI Stream).

Відносно невелика вартість реалізації подібних обчислювальних кластерів робить їх привабливими для наукових потреб вищих навчальних закладів та науково-дослідних інституцій.

Програмні платформи

Впродовж останніх семи років традиційною програмною платформою для обчислювальних кластерів є різновиди ОС Linux [17]. Ця ОС впевнено займає позиції лідера у сфері високопродуктивних обчислень завдяки своїй гнучкості, кросплатформеності, відкритості вихідних кодів, стабільності та надійним засобам безпеки. Станом на листопад 2011 року у списку Top 500 майже 92% кластерів побудовано на платформі ОС Linux.

Починаючи з 2003 року, у Top 500 з'являються кластери на платформі ОС сімейства Microsoft Windows. Для побудови обчислювального кластера із вузлами під керуванням Windows застосовувався програмний пакет Windows Compute Cluster Server 2003, на зміну якому згодом прийшов Windows HPC Server 2008. У порівнянні з Linux-кластерами, такі обчислювальні кластерні системи можуть бути побудовані на базі більш обмеженої кількості можливих апаратних платформ, вони більш вимогливі до апаратних ресурсів, характеризуються значно вищою вартістю необхідного для функціонування ПЗ. За продуктивністю кластери на платформі Windows майже не поступаються Linux кластерам [20–23]. Серед переваг Windows кластера можна виділити такі:

• можливість інтеграції з існуючою інфраструктурою Active Directory, робочих станцій і серверів під керуванням ОС Microsoft Windows;

• можливість створення гібридних обчислювальних кластерів на платформах Windows та Linux;

• можливість виконання обчислень у таблицях популярного пакета Microsoft Excel на кластері;

• можливість виконання розрахунків на відеоадаптерах за допомогою інтерфейсу DirectX 11.

На сьогоднішній день науковою спільнотою проведено недостатню кількість досліджень обчислювальних кластерів на платформі Microsoft Windows. Зважаючи на той

факт, що системне ПЗ для таких кластерів є безкоштовним для наукових цілей вищих навчальних закладів, можна із впевненістю говорити про актуальність проведення наукових досліджень у даному напрямі.

3. Сучасні технології апаратно-програмної віртуалізації

У широкому сенсі поняття віртуалізації являє собою приховування від користувача справжньої реалізації будь-якого процесу або об'єкта. З появою експериментального зразка машини IBM M44/44X вперше було вжито термін «віртуальна машина» (virtual machine), який замінив більш ранній термін «псевдомашина» (pseudo machine). Згодом у мейнфреймах серії IBM System 360/370 можна було використовувати ВМ для емуляції попередніх версій ОС, було реалізовано концепцію супервізора, що забезпечував відокремлення розділів і контролював доступ задач до ресурсів [24, 25]. Але лише у 90-х роках стали очевидними перспективи підходу віртуалізації, оскільки зі зростанням апаратних потужностей як персональних комп'ютерів, так і серверних рішень, з'явилася можливість використовувати декілька ВМ на одній фізичній платформі.

Поняття віртуалізації умовно можна розділити на дві категорії, що фундаментально різняться:

• віртуалізація платформ: продуктом цього виду віртуалізації є ВМ – програмні абстракції, що запускаються на платформі реальних апаратно-програмних систем;

• віртуалізація ресурсів – комбіноване чи спрощене уявлення апаратних ресурсів для користувача і отримання деяких користувацьких абстракцій обладнання, просторів імен, мереж тощо.

Під віртуалізацією платформ розуміють створення програмних систем на основі існуючих апаратно-програмних комплексів, які залежать або не залежать від них. Система, що надає апаратні ресурси і ПЗ, називається сервером віртуальних машин (СВМ), а системи, що нею симулюються, – гостьовими (guest). Для стабільного функціонування гостьових систем на платформі СВМ необхідно, щоб ПЗ та АЗ СВМ надавали необхідний набір інтерфейсів для доступу до його ресурсів. Є кілька варіантів віртуалізації платформ, у кожному з яких здійснюється свій підхід до поняття «віртуалізація». Кожний варіант відрізняється тим, наскільки повно здійснюється симуляція апаратного забезпечення. Види віртуалізації платформ представлено нижче [26].

• *Повна емуляція*. При такому виді віртуалізації ВМ повністю віртуалізує все АЗ зі збереженням гостьової ОС в незмінному вигляді. Такий підхід дозволяє емулювати різні апаратні архітектури. Прикладами ВМ, що працюють на даному принципі, є Microsoft Virtual PC, VMware Workstation, GXemul, OVPsim, QEMU, QuickTransit, SIMH.

• *Часткова емуляція*. У цьому випадку ВМ віртуалізує лише необхідну кількість АЗ для можливості ізольованого запуску ОС. Такий підхід дозволяє запускати гостьові ОС, розроблені тільки для тієї ж архітектури, що й у СВМ. Цей вид віртуалізації дозволяє істотно збільшити продуктивність гостьових систем у порівнянні з повною емуляцією і широко використовується на цей час. З метою підвищення швидкодії в платформах віртуалізації, які використовують даний підхід, застосовується спеціальний «прошарок» між гостьовою ОС та обладнанням – гіпервізор (hypervisor), або «монітор віртуальних машин» (Virtual Machine Monitor, VMM), що дозволяє гостьовій системі безпосередньо звертатися до ресурсів АЗ. Застосування гіпервізора, що є сполучною ланкою між гостьовими системами та АЗ, істотно збільшує швидкодію, наближаючи її до швидкодії фізичної платформи. Прикладами таких систем є VMware ESX Server, Microsoft Hyper-V Server 2008, Virtual Iron, Sun xVM.

• *Часткова віртуалізація*, а також «віртуалізація адресного простору» («address space virtualization»). При такому підході ВМ симулює кілька примірників апаратного оточення (але не всього), зокрема, адресні простори. Такий вид віртуалізації дозволяє

спільно використовувати ресурси та ізолювати процеси, але не дозволяє розділяти екземпляри гостьових ОС. Іншими словами, користувачем не створюються ВМ, а відбувається ізоляція будь-яких процесів на рівні ОС. Ці механізми реалізовано в усіх сучасних ОС.

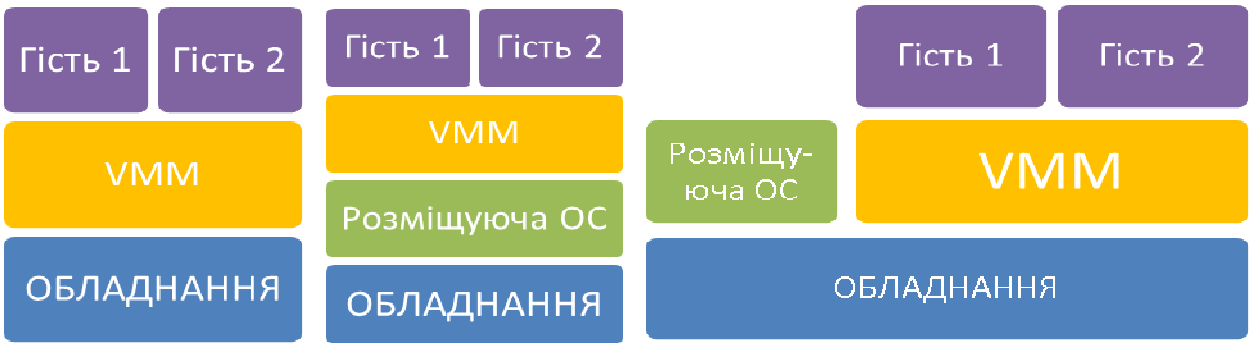
• *Паравіртуалізація* (Paravirtualization). При застосуванні паравіртуалізації немає необхідності емулювати АЗ. Однак замість цього (або додатково до цього) ядро гостьової ОС модифікується так, що виклики привілейованих інструкцій замінюються на виклики функцій прикладного програмного інтерфейсу (API) гіпервізора. Таким чином, системи для паравіртуалізації також мають свій гіпервізор, а API-виклики до гостьової системи називаються гіпервикликами (hypercalls). Прикладами ВМ, що працюють за принципом паравіртуалізації, є Xen, XtratuM, Wind River hypervisor.

• *Віртуалізація рівня ОС*. Суттю такого виду віртуалізації є віртуалізація фізичного сервера на рівні ОС з метою створення декількох захищених віртуалізованих серверів на одному фізичному. Гостьова система в даному випадку розділяє використання одного ядра ОС СВМ з іншими гостьовими системами. ВМ являє собою оточення для застосувань, що запускаються ізольовано. Такий тип віртуалізації застосовується при організації систем розміщення, коли в рамках одного примірника ядра потрібно підтримувати декілька віртуальних серверів клієнтів. Приклади систем із віртуалізацією рівня ОС: OpenVZ, Linux-VServer, Virtuozzo, KVM.

• *Віртуалізація рівня застосувань*. У даному випадку застосування поміщається в «контейнер» з необхідними елементами для своєї роботи: файлами реєстру, призначеними для користувача файлами налаштувань, системними об'єктами. Віртуальне оточення, створене для програми, розв'язує конфлікти між нею і ОС, а також іншими застосуваннями. Прикладами таких систем є ВМ Java, яка стала стандартом де-факто для мобільних платформ, та платформа .NET.

Сучасні архітектури віртуалізації. Типи гіпервізорів

До виділених Р. Голдбергом у 1970-х роках [26, 27] двох узагальнених архітектур віртуалізації (рис. 1 а, рис. 2 б) можна додати ще одну, яка містить риси як першої, так і другої архітектури (рис. 1 в). Фундаментальна відмінність між ними стосується сфери відносин між рівнем віртуалізації і фізичним обладнанням. Під рівнем віртуалізації розуміють рівень монітора ВМ. Саме цей рівень надає можливість створювати кілька ізольованих примірників на основі одних і тих самих апаратних ресурсів.



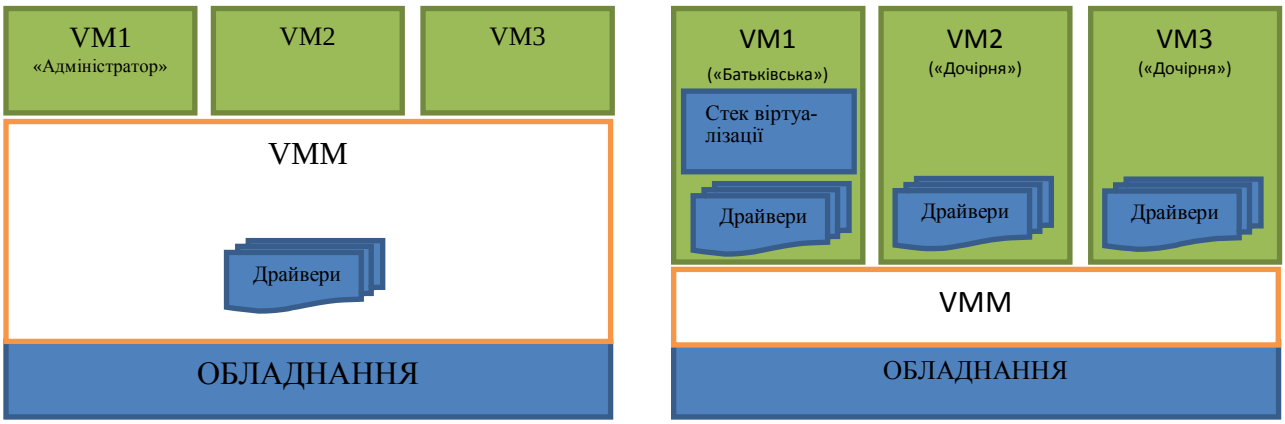
а) VMM типу 1 б) VMM типу 2 в) гібридний VMM

Рис. 1. Архітектури віртуалізації

У випадку архітектури VMM типу 2 гіпервізор, запущений разом з ОС СВМ, дозволяє створювати ВМ. При такій архітектурі запити гостьових систем до апаратних ресурсів обслуговуються розміщуючою ОС, а ВМ зазвичай є її процесами. Прикладами таких VMM є Microsoft Virtual PC, VMware Workstation. Варто зазначити, що при такій

архітектурі VMM додає істотні накладні витрати і, отже, не підходить для робочих навантажень, що інтенсивно використовують ресурси.

В архітектурі VMM типу 1 рівень VMM працює безпосередньо над обладнанням. Ця архітектура вперше була розроблена фірмою IBM у 1960-ті роки для мейнфреймів і нещодавно стала доступною на платформах x86/x64. У VMM типу 1 додатково можна виділити два основні підтипи архітектур: мікроядерний та монолітний (рис. 2).



а) монолітний гіпервізор б) мікроядерний гіпервізор

Рис. 2. Основні архітектури гіпервізорів

Монолітний підхід розміщує гіпервізор в єдиному рівні, який також включає більшість необхідних компонентів, таких як ядро, драйвери пристроїв, стек вводу/виводу. Цей підхід використовується такими рішеннями, як VMware ESX, Xen, і традиційними системами мейнфреймів.

Мікроядерний підхід використовує «тонкий», спеціалізований гіпервізор, який виконує лише основні завдання забезпечення ізоляції розділів та керування пам'яттю. Цей рівень не включає стека вводу/виводу та драйверів пристроїв і використовується у Microsoft Hyper-V. У цій архітектурі стек віртуалізації і драйвери конкретних пристроїв розташовані в ОС СВМ – спеціальному розділі, який називають батьківським розділом.

Гібридний VMM (або віртуалізація з розміщенням [28]) працює на одному і тому самому рівні привілеїв, що і розміщуюча ОС. При такому підході гіпервізор надає гостьовим системам доступ до ресурсів безпосередньо, без втручання ОС СВМ, але використовує функції ядра розміщуючої ОС, які сам не реалізує (планувальник черги завдань, керування пам’яттю, стек вводу-виводу, керування живленням тощо). Прикладами систем, побудованих за даним принципом, є Microsoft Virtual Server, KVM. Дещо схожий підхід використовується у так званій віртуалізації на основі контейнерів (container-based virtualization). Тут примірники гостьових ОС спільно використовують ядро розміщуючої ОС, але мають власні набори процесів і простори адрес. На відміну від гібридного підходу, віртуалізація на основі контейнерів взагалі не використовує гіпервізор. Особливістю такої технології є обмеженість варіантів можливих гостьових ОС такими, що мають спільне ядро з розміщуючою. Прикладами систем є OpenVZ, Linux-VServer, Virtuozzo.

Принципи функціонування апаратно-програмної віртуалізації

Для підтримки апаратно-програмної віртуалізації архітектуру сучасних мікропроцесорів (МП) було дещо змінено за рахунок введення додаткових інструкцій з метою надання прямого доступу до ресурсів процесора із гостьових систем. Цей набір додаткових інструкцій носить назву Virtual Machine Extensions (VMX). Підтримка віртуалізації на рівні

МП забезпечується новим режимом роботи, який в термінології Intel називається роботою в режимі VMX. VMX має такі інструкції [29]:

• VMPTRLD – завантаження вказівника на структуру керування ВМ (Virtual Machine Control Structure, VMCS);

• VMPTRST – збереження вказівника на VMCS у комірці пам'яті;

• VMCLEAR – очищення VMCS;

• VMREAD – зчитування значення певного поля VMCS;

• VMWRITE – запис у поле VMCS;

• VMCALL – запит до сервісних функцій гіпервізора;

• VMLAUNCH – виконання ПЗ ВМ, що керується поточною VMCS у «гостьовому» режимі;

• VMRESUME – виконання ПЗ ВМ, що керується поточною VMCS у «гостьовому» режимі. Призначена для поновлення роботи ВМ після тимчасової передачі управління гіпервізору;

• VMXON – увімкнення режиму роботи VMX;

• VMXOFF – вимкнення режиму роботи VMX.

Виділяють два типи роботи в режимі VMX: VMX root operation і VMX non-root operation. Як правило, монітор ВМ працює в режимі root, а "гостьове" ПЗ – в режимі non-root. Переходи між режимами VMX root і VMX non-root називаються "входами VMX", а переходи між VMX non-root і VMX root – "виходами VMX".

Поведінка МП в режимі VMX root мало відрізняється від звичайної роботи. Принципова різниця полягає в доступності нового набору інструкцій VMX і в обмеженні на значення, які можуть бути завантажені в певні керуючі регістри. Поведінка процесора в режимі VMX non-root характеризується низкою обмежень і змін у порівнянні зі звичайним: набір спеціальних інструкцій ВМ заміняє деякі із стандартних інструкцій. Функціональність ПЗ, що працює в режимі non-root, обмежена. Це зроблено для того, щоб VMM міг зберігати повний контроль над ресурсами процесора.

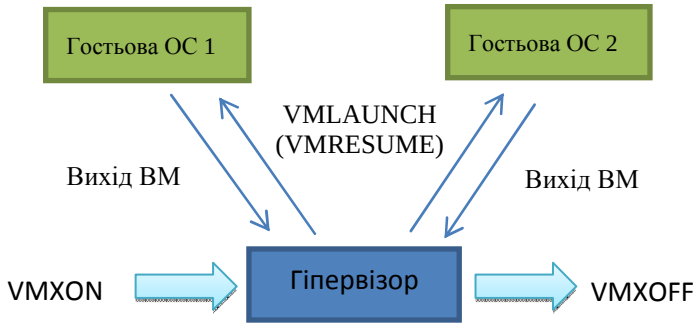


Рис. 3. Цикл роботи гіпервізора

На рис. 3 зображено схему циклу роботи VMM і "гостьового" ПЗ, яка показує взаємовідносини між цим ПЗ і командами модифікації режимів. Програма починає виконуватися в режимі VMX за допомогою виконання інструкції VMXON. Після цього VMM може "запросити" своїх "гостей" у ВМ (по черзі) за допомогою входів VMX. Монітор ВМ виконує вхід VMX за допомогою інструкцій VMLAUNCH і VMRESUME і повертає собі управління за допомогою виходу VMX. Вихід VMX передає управління в точці, зазначеній VMM. Монітор ВМ може вживати дії відповідно до причин виходу VMX, а потім може знову повернутися до ВМ через вхід VMX. Нарешті, в моніторі ВМ може бути прийнято рішення про вимикання і вихід з роботи в режимі VMX. Ця дія здійснюється за допомогою інструкції VMXOFF.

Порівняльний аналіз сучасних платформ апаратно-програмної віртуалізації

Донедавна ВМ не розглядалися як потенційні вузли обчислювальних кластерів через значні накладні витрати на віртуалізацію. Широке розповсюдження архітектури мікропроцесора x64 та поява у сучасних процесорах засобів апаратної віртуалізації дозволили змінити ставлення до продуктивності віртуальних платформ у порівнянні із

фізичною. Наближення продуктивності ВМ до продуктивності невіртуалізованих апаратних засобів сприяло спрямуванню поглядів наукової спільноти у напрямі побудови обчислювальних кластерів на ВМ, гіпервізори яких використовують відповідні розширення системи команд центрального МП [30–34].

У деяких роботах пропонується використовувати контейнерну віртуалізацію як альтернативу гіпервізорам для високопродуктивних обчислень [32, 35]. Перевагами такого підходу є відсутність накладних витрат на повну ізоляцію гостьових систем, що дозволяє дещо пришвидшувати операції вводу-виводу і звільняти більше ОП для робочих навантажень. Вагомим недоліком такого підходу є неможливість роботи гостьових систем, ядра яких відмінні від ядра розміщуючої ОС. Таким чином, за допомогою даної технології неможливо побудувати гетерогенні за ОС вузла кластери (наприклад, сумістити Windows і Linux кластери).

Найбільш пріоритетним напрямком для виробників платформ віртуалізації є віртуалізація серверів. Це не дивно, оскільки підвищення керованості, відмовостійкості, розподіл навантаження, масштабованість та оптимізація використання апаратних ресурсів, яку можуть дати ВМ, найбільш бажані при наданні послуг розміщення (веб-сайти, бази даних тощо). Вдосконалення платформ віртуалізації робочих навантажень у напрямку продуктивності та доступності (відмовостійкості) є пріоритетами виробників таких систем.

Порівняння функцій та можливостей найуживаніших на сьогоднішній день технологій віртуалізації серверів, які працюють на базі гіпервізорів, наведено у табл. 1 [36–40]. Тут прийнято до уваги, що архітектурою процесора як СВМ, так і ВМ, є x64 – архітектура сучасних ПК. Порівняння проведено за такими критеріями і параметрами:

• принцип роботи – який із принципів віртуалізації використовується у платформі;

• типове використання – який сценарій використання платформи найбільш відповідає її функціональним можливостям;

• продуктивність відносно апаратної платформи – продуктивність ВМ відносно ПК з аналогічною конфігурацією;

• жива міграція – можливість перенесення ВМ без переривання роботи на інший СВМ;

• призупинення працюючої системи – миттєве призупинення роботи ВМ на довільний час з можливістю миттєвого її відновлення;

• динамічне виділення ОП – виділення ВМ необхідного об’єму ОП з наявної вільної ОП СВМ в залежності від потреб ВМ у процесі роботи;

• максимальна можлива кількість ВМ на СВМ – максимальна дозволена платформою кількість одночасно працюючих ВМ на одному СВМ;

• максимальна кількість віртуальних процесорів на ВМ – максимальне допустиме значення кількості віртуальних процесорів (або ядер) ВМ;

• максимальний об’єм ОП на ВМ – максимальне припустиме значення об’єму ОП ВМ за умови необмеженого об’єму доступної ОП СВМ;

• ОС на СВМ – ОС, що працює на СВМ одночасно із гостьовими ОС для підтримання роботи гіпервізора і платформи в цілому;

• доступність повнофункціональної безкоштовної версії для навчальних цілей вищих навчальних закладів (ВНЗ). Тобто можливість ВНЗ безкоштовно отримати копію ПЗ з повним набором функцій, необхідних для проведення відповідних досліджень, на необхідний для таких цілей час.

Згідно з цією таблицею, сучасні технології віртуалізації, які можуть бути застосовані для високопродуктивних обчислень, мають майже однакові функції відносно управління ресурсами і засобів відмовостійкості. Важливим показником для можливості проведення досліджень є наявність повнофункціональних безкоштовних версій.

Таблиця 1. Основні функції та можливості найбільш вживаних технологій віртуалізації серверів

	Microsoft Hyper-V Server 2008 R2	KVM	Virtual Iron 4.5	VMware ESX Server (vSphere) 4.0	Xen
Принцип роботи	Часткова емуляція	Віртуалізація рівня ОС	Часткова емуляція	Часткова емуляція	Паравіртуалізація
Типове використання	Консолідація серверів, розробка, тестування, приватні «хмари»	Консолідація серверів	Консолідація серверів, розробка, тестування, відмовостійкість	Консолідація серверів, розробка, тестування, відмовостійкість, приватні «хмари»	Консолідація серверів, розробка, тестування
Продуктивність відносно апаратної платформи	Майже співпадає	Майже співпадає	Майже співпадає	Майже співпадає	Майже співпадає
Жива міграція	+	+	+	+	+
Призупинення працюючої системи	+	+	+	+	+
Динамічне виділення ОП	+	+	-	+	+
Максимальна можлива кількість ВМ на СВМ	384	8 * кількість ядер на СВМ	-	320	Визначається ОС СВМ та ВМ
Максимальна кількість віртуальних процесорів на ВМ	4	16	Кількість процесорів на СВМ	8	Визначається ОС СВМ та ВМ
Максимальний об'єм ОП на ВМ	64 GB	512 ГБ	Визначається апаратною архітектурою СВМ	255 GB	Визначається ОС СВМ та ВМ
Операційна система на СВМ	Microsoft Windows Server 2008 R2, Microsoft Windows Server 2008 R2 Server Core	Linux	-	-	NetBSD, Linux, Solaris
Доступність повнофункціональної безкоштовної версії для навчальних цілей ВНЗ	+	+	-	-	+

Саме це є однією з причин того, що ВМ на платформі Linux найчастіше застосовуються і вивчаються. Зазвичай при виборі платформи віртуалізації вагомими

чинниками також є часова вартість розгортання і підтримки, можливості інтеграції з існуючою інфраструктурою, наявність зручних API взаємодії з платформою для автоматизації досліджень тощо. В подальших розділах буде розглянуто концепцію побудови обчислювального кластера на платформі ОС Microsoft Windows Server 2008 R2 із застосуванням технології віртуалізації Microsoft Hyper-V Server 2008 R2.

4. Застосування ВМ як вузлів обчислювального кластера з гнучкою архітектурою на платформі Microsoft Hyper-V

Розгортання і експлуатація обчислювального кластера на ВМ принципово не відрізняється від аналогічних операцій на ПК. Віртуалізація до функціональності кластера додає низку можливостей програмної взаємодії із ВМ, які також можна використати для впливу на процеси виконання обчислювальних завдань на рівні обчислювальних вузлів, а не на рівні завдань як таких.

Незважаючи на схожість у наборі реалізованих функцій, кожна платформа апаратно-програмної віртуалізації побудована по-різному. Тому підходи до розгортання таких систем, взаємодії з ними можуть докорінно відрізнятися в залежності від платформи. Особливо чітко це проявляється при використанні API платформи. Так, наприклад, Xen API ґрунтується на віддаленому виклику процедур, параметрами яких є рядкові дані у форматі XML (Extensible Markup Language) [39]; KVM API являє собою системні виклики під назвою ioctl (input/output control), вхідними і вихідними параметрами яких є дескриптори файлів [38]; Microsoft Hyper-V API представлено класами WMI (Windows Management Instrumentation) [41, 42]. Не менш вагомими є відмінності у вимогах до апаратно-програмної конфігурації СВМ, оскільки це впливає як на процес розгортання, так і на процес експлуатації системи. Так, для реалізації «живої» міграції ВМ на платформі KVM всі СВМ повинні знаходитися в одній і тій самій локальній підмережі та мати доступ до деякого спільного мережевого сховища з образами ВМ, наприклад, на базі технологій SAN (Storage Area Network), NAS (Network Attached Storage). ВМ на платформі Xen для тих самих цілей підходять лише SAN-сховища. Для налаштування «живої» міграції ВМ на платформі Microsoft Hyper-V Server R2 відповідні СВМ повинні бути об’єднаними у так званий відмовостійкий кластер (Failover Cluster) з приєднаним спільним томом (Cluster Shared Volume), що являє собою SAN-сховище (наприклад, Internet Small Computer System Interface (iSCSI), Fiber Channel).

Таким чином, враховуючи вищенаведені факти, реалізація системи та процес її дослідження може докорінно відрізнятися для різних платформ. Реалізовані функції API можуть вплинути на алгоритми та методики проведення досліджень. Подальша частина роботи буде пов’язана саме з платформою Microsoft Hyper-V R2.

Застосування ВМ на платформі Microsoft Hyper-V Server 2008 R2 як вузлів обчислювального кластера

Концепцію створення гнучких гомогенних архітектур кластерних систем було висунуто в [8], а далі поглиблено до побудови динамічно реконфігурованої кластерної обчислювальної системи на базі платформи віртуалізації Microsoft Hyper-V [41]. Там же описані вимоги до апаратно-програмної конфігурації СВМ. У вимогах не було враховано необхідні конфігурації для реалізації міграції ВМ (оскільки вони сформульовані для попередньої версії платформи – Microsoft Hyper-V Server 2008, в якій механізм міграції був відсутнім), тому додатково необхідно врахувати вимоги до апаратно-програмної конфігурації відмовостійкого кластера [43]. Таким чином, при даній конфігурації ВМ фактично представлена у двох варіантах: у вигляді гостьової системи і у вигляді сервісу відмовостійкого кластера.

Управління ресурсами кластера інфраструктурою керування

Під обчислювальними ресурсами кластера будемо розуміти апаратні ресурси, конфігурація яких суттєво впливає на можливість і продуктивність виконання обчислень. Іншими словами, конфігурація обчислювального кластера складається із множини конфігурацій його вузлів та мережевого з'єднання між ними. В свою чергу, апаратна конфігурація кожного вузла – це об'єм його ОП, параметри центрального і допоміжних (при наявності) МП, характеристики накопичувача на жорсткому диску тощо. У випадку віртуальних платформ ці ресурси фактично є деякими програмними абстракціями, що надає гнучкості оперування ними в межах наявних фізичних апаратних ресурсів.

Згідно з [41, 42], інфраструктура керування ВМ являє собою набір класів WMI. Класи, що відповідають за роботу ВМ як гостьової системи, знаходяться у просторі імен WMI Root\Virtualization, а класи, що відповідають за ВМ, у складі відмовостійкого кластера у просторі імен Root\MSCluster. Аналіз API-платформи віртуалізації та API-пакета Microsoft HPC Server 2008 дозволив сформулювати набір операцій, які можуть бути виконані для управління ресурсами кластера. Ці операції наведено у табл. 2.

Таблиця 2. Засоби управління ресурсами кластера

№	Функція	Потребує перезапуску задачі	Межі застосування	Чим управляє; на що впливає
1	Модифікація об'єму ОП (статична ОП)	+	8 – R_a[1] [МБ]	Об'єм пам'яті, доступної для задачі/ОС СВМ. Кількість можливих ВМ на СВМ
2	Модифікація кількості процесорів	+	1 – 4 шт.	Кількість паралельних процесорів, доступних для задачі; перерозподіл процесорного часу між ВМ
3	Призначення вузла NUMA	+	ВМ	Продуктивність роботи процесора і пам'яті
4	Модифікація типу ОП (статична, динамічна)	+	ВМ	Тип розподілу ресурсів ОП між ВМ у межах одного СВМ; кількість можливих ВМ на СВМ
5	Автоматична зміна об'єму ОП (динамічна ОП)	–	ВМ	Об'єм пам'яті, доступної для задачі/ОС СВМ; кількість можливих ВМ на СВМ
6	Жива міграція	–	Відмовостійкий кластер	Розподіл ВМ по СВМ. Швидкість мережевого зв'язку між ВМ
7	Швидка міграція	Залежить від задачі	Відмовостійкий кластер	Розподіл ВМ по СВМ. Швидкість мережевого зв'язку між ВМ
8	Модифікація пріоритетів щодо процесорного часу	–	1 – 100 відносних одиниць	Виділення процесорного часу ВМ

[1] R_a – об'єм доступної ОП на СВМ.

Продовж. табл. 2

9	Модифікація пріоритетів динамічної ОП	–	1 – 10 відносних одиниць	Виділення ресурсів ОП ВМ
10	Розподіл задач за власним законом	+	Обчислювальний кластер	Порядок виділення обчислювальних ресурсів (процесорів та вузлів) задачам
11	Встановлення пріоритетів задач	+	5 ступенів пріоритетності	Місце задачі у черзі за отриманням ресурсів

Розглянемо більш детально зміст наведених операцій, попередньо згрупувавши їх за ресурсами.

• Центральний процесор (ЦП) (операції 2, 3, 8). ЦП ВМ не мають взаємно однозначної відповідності із МП чи ядрами СВМ, а разом із пріоритетами визначають частину загального процесорного часу, що може бути отриманим ВМ, і, звісно, кількість паралельних процесорів, доступних ОС ВМ. У випадку багатопроцесорних СВМ використовується архітектура NUMA. Тобто, кожен процесор працює із певною частиною ОП і таким чином формується вузол NUMA. Налаштування за замовчанням передбачає автоматичне призначення ВМ вузла NUMA. У випадку декількох ВМ на одному СВМ таке призначення може виявитися не досить оптимальним. Тому примусове призначення відповідного вузла може значно покращити продуктивність.

• ОП (операції 1, 4, 5, 9). ОП ВМ характеризується декількома параметрами, основними з яких є об’єм для запуску ВМ, максимально можливий об’єм (для динамічної ОП) і пріоритет ОП (для динамічної ОП). ВМ не може бути запущена, якщо на СВМ об’єм доступної ОП не перевищує об’єм ОП для запуску відповідної ВМ.

• Конфігурація мережевого з’єднання між ВМ (операції 6, 7). Для розподілених паралельних задач, що обмінюються даними між вузлами (наприклад, MPI), пропускна спроможність (швидкість) та затримки мережевого з’єднання є важливими параметрами, які вагомо можуть вплинути на продуктивність обчислень. Швидкість мережевого з’єднання між ВМ у межах СВМ становить 10 Гбіт/с, у той час, як ПК зазвичай з’єднуються мережею 100 Мбіт/с чи 1 Гбіт/с. Таким чином, розташовуючи відповідні ВМ на одному і тому самому СВМ, можемо досягти підвищення продуктивності виконання обчислень за рахунок швидкості мережевого з’єднання. Без вимикання ВМ це може бути реалізовано операціями 6, 7. Швидка міграція відрізняється від живої тим, що ВМ призупиняє роботу на час процесу міграції і потім знову відновлює попередній стан.

Окреме місце посідають операції над задачами обчислювального кластера. Вони не пов’язані із ресурсами ВМ, оскільки виконуються засобами програмного пакета для створення кластера (у нашому випадку це Microsoft HPC Server 2008). Ці операції не змінюють ресурси кластера, а дозволяють вплинути на процес їх використання.

Динамічна реконфігурація ресурсів кластера. Приклади можливих конфігурацій

Операції 1–9, наведені в табл. 2, дозволяють здійснювати динамічну реконфігурацію ресурсів обчислювального кластера. Слово «динамічна» використано у сенсі «без внесення змін до фізичної апаратної платформи». Приклади можливих конфігурацій наведено на рис. 4.

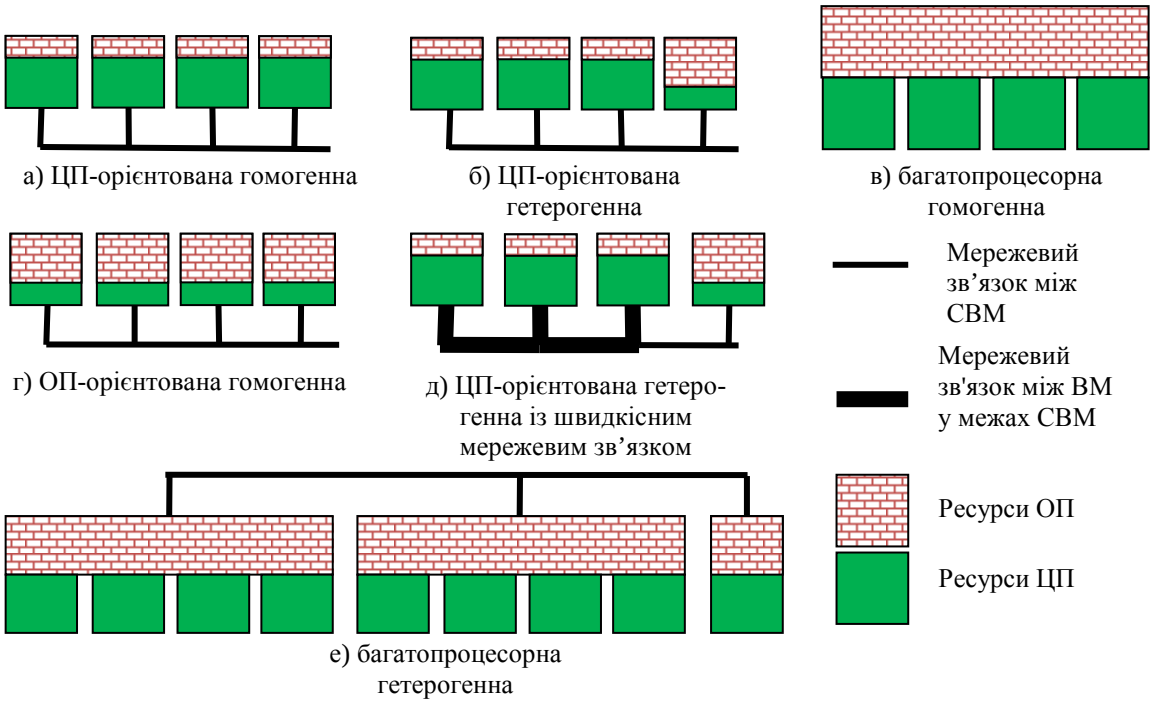


Рис. 4. Приклади можливих конфігурацій

5. Оптимізація використання ресурсів обчислювального кластера

Для отримання всіх переваг, які надає віртуалізація обчислювальному кластеру, необхідні засоби оптимального завантаження ресурсів та зручного управління системою ВМ. Під оптимізацією розуміють мінімізацію чи максимізацію певного функціоналу. Традиційно оптимізацію завантаженості ресурсів обчислювального кластера розв’язують за допомогою алгоритмів планування задач [44], мінімізуючи змагання задач за ресурси, час очікування в черзі тощо. Віртуалізація надає можливість консолідації і перерозподілу навантажень, що дозволяє оптимізувати завантаження ресурсів на більш низькому рівні в режимі реального часу. В цьому випадку мінімізації може підлягати, наприклад, імовірність виходу з ладу обчислювальної системи або частини її вузлів, завантаженість певних обчислювальних ресурсів, спожита електроенергія [45]. В цій роботі пропонується використовувати завантаженість ЦП СВМ і ВМ як метрику для розв’язання задачі оптимізації.

Коли говорять про продуктивність виконання обчислень, то розуміють кількість виконаних операцій за певний проміжок часу. Ця величина може залежати як від характеристик апаратних засобів, так і від взаємодії між ними. Наприклад, незалежно від обчислювальної потужності ЦП, у випадку нестачі ОП, ОС і її процеси починають використовувати файли підкачки. В результаті збільшується час простою ЦП, що значно знижує загальну продуктивність. З іншого боку, якщо розмістити всі процеси MPI задачі в межах одного вузла з багатоядерним ЦП, то це може вагомо збільшити швидкість обрахунків.

Для визначення нестачі ресурсів користуються метриками, що відповідають цим ресурсам. Але в будь-якому випадку вагома нестача обчислювальних ресурсів призводить до падіння завантаженості ЦП і, навпаки, завантаженість ЦП зростає при сприятливих умовах для задачі. Таким чином, показники завантаженості ЦП вузлів є індикатором сприятливості конфігурації обчислювальної системи для задачі, що виконується. Отже, можливо сформулювати задачу оптимізації конфігурацій кластера за параметром завантаженості ЦП його вузлів.

Задача оптимізації завантаженості ресурсів кластера за допомогою керування ресурсами ВМ може бути сформульована таким чином.

Знайти z за умови:

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

Тут $F(z)$ – сумарна завантаженість процесорів усіх СВМ, $H_i(z)$ – сумарна завантаженість процесорів усіх ВМ на СВМ з індексом i, L_{ik} – завантаженість k-го ядра процесора i-го СВМ, N – кількість СВМ.

Керування ресурсами ВМ є гнучким механізмом, але він не дозволяє розв'язати деякі важливі супутні технічні задачі. Платформа віртуалізації не оперує параметрами запуску задач на кластері, а навіть при готовій оптимальній конфігурації буде неможливо запустити відповідні процеси на відповідних вузлах кластера без механізмів керування розподілом задач. Таким чином, цей механізм доцільно застосовувати у поєднанні із керуванням ресурсами ВМ.

6. Висновки

1. Висунуту концепцію створення гнучких гомогенних архітектур кластерних систем поглиблено за рахунок можливості формування динамічно реконфігурованої кластерної обчислювальної системи з використанням механізмів віртуалізації платформи Microsoft Hyper-V.
2. На основі виконаного аналізу технологій віртуалізації при побудові кластерних обчислювальних систем з гнучкою архітектурою показано актуальність використання апаратної платформи ПК і серверів для реконфігурації ресурсів обчислювальних кластерів.
3. Проведений аналіз сучасних платформ апаратно-програмної віртуалізації дозволив сформулювати актуальність використання і дослідження ВМ на платформі Microsoft Hyper-V R2 як вузлів обчислювального кластера.
4. На основі аналізу функціональних можливостей API платформи Microsoft Hyper-V R2 сформульовано задачу оптимізації обчислювальних ресурсів кластера.

СПИСОК ЛІТЕРАТУРИ

1. Филиппов А.К. Теоретические основы проектирования динамически реконфигурируемых систем обработки информации: учебн. пособ. / Филиппов А.К. – Владимир: Изд-во Владим. гос. ун-та, 2009. − 119 с.
2. Иванов А.И. О создании и применении проблемно-ориентированных комплексов, предназначенных для решения задач, обладающих высокой емкостной и временной сложностью / А.И. Иванов // Штучний інтелект. − 2004. − № 4. − С. 15 – 26.
3. Гузик В.Ф. Проблемно-ориентированные высокопроизводительные вычислительные системы: учебн. пособ. / В.Ф. Гузик, В.Е. Золотовский. − Таганрог: Изд-во ТРТУ, 1998. – 236 с.
4. Опанасенко В.Н. Высокопроизводительные реконфигурируемые компьютеры на базе FPGA / В.Н. Опанасенко // Проблеми інформатизації та управління: зб. наук. праць НАУ. – 2009. – Вип. 3 (27). – С. 114 – 118.
5. Палагин А.В. Проблемная ориентация в развитии компьютерных архитектур / А.В. Палагин, А.Ф. Кургаев // Кибернетика и системный анализ. − 2003. − № 4. − С. 167 − 180.
6. Палагин А.В. Об ЭВМ с виртуальной архитектурой / А.В. Палагин // Управляющие системы и машины. − 1999. − № 3. − С. 33 – 43.

7. Развитие аппаратной платформы реконфигурируемых вычислительных систем / Н.Н. Дмитриенко, И.А. Каляев, И.И. Левин [и др.] // Научный сервис в сети Интернет: суперкомпьютерные центры и задачи. Труды междунар. суперкомпьютерной конф., (Новороссийск, 20−25 сентября 2010 г.). − Новороссийск, 2010. − С. 315 – 320.
8. 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.
9. Білоконь І. Створення захищеного обчислювального windows-кластера на платформі desktop PC / І. Білоконь, Д. Грязнов, В. Мар'яновський // Вісник Київського національного університету імені Тараса Шевченка. – (Серія «Радіофізика та електроніка»). – 2009. – Вип. 12. – С. 13 – 16.
10. Beowulf cluster [Електронний ресурс]. – Режим доступу: http://www.beowulf.org.
11. Windows HPC Server 2008 R2 Technical Library [Електронний ресурс]. – Режим доступу: http://technet.microsoft.com/en-us/library/ee783547%28WS.10%29.aspx.
12. C.mmp –A multi-mini-processor / W.A. Wulf, C.G. Bell // Proc. of the Fall Joint Computer Conference. – New York, 1972. – P. 765 – 777.
13. Parallel Virtual Machine [Електронний ресурс]. – Режим доступу: http://www.csm. ornl.gov/pvm.
14. Windows Azure [Електронний ресурс]. – Режим доступу: http://www.windowsazure.com/ru-ru/home/tour/overview.
15. Windows HPC Server and Windows Azure. High-Performance Computing in the Cloud: (Whitepaper) [Електронний ресурс] / D. Chappell // Microsoft Corporation. – 18 p. – Режим доступу: http://download.microsoft.com/download/E/E/C/EEC2AC07-5E0B-4D7C-858E-C60FC9C60EB8/ Windows%20HPC%20Server%202008%20R2%20SP2%20and%20Windows% 20Azure.pdf.
16. Amazon Elastic Compute Cloud [Електронний ресурс]. – Режим доступу: http://aws.amazon.com/ec2.
17. Top 500 [Електронний ресурс]. – Режим доступу: http://top500.org.
18. Погорілий С.Д. Використання технології GPGPU для збільшення швидкодії гомогенних кластерних систем / С.Д. Погорілий, М.І. Трибрат, Д.Ю. Вітель // Теоретичні та прикладні аспекти побудови програмних систем (TAAPSD'2011): VIII міжнар. наук.-практ. конф., тезиси допов., (Ялта, 19–23 вер. 2011 р.). – Ялта, 2011. – С. 171 –176.
19. Анализ методов повышения производительности компьютеров с использованием графических процессоров и программно-аппаратной платформы CUDA / С.Д. Погорелый, Ю.В. Бойко, М.И. Трибрат [и др.] // Математичні машини і системи. – 2010. – № 1. – С. 40 – 54.
20. A Hybrid Operating System Cluster Solution: Dual-Boot and Virtualization with Windows HPC Server 2008 and Linux Bull Advanced Server for Xeon: (Whitepaper) [Електронний ресурс] / Microsoft Corporation. – 76 p. – Режим доступу: http://go.microsoft.com/ fwlink/?LinkId=164839.
21. Performance comparison of scientific applications on Linux and Windows HPC Server clusters / Albino A. Aveleda, Renato N. Elias, José J. Camata [et al.] // Mecánica Computacional. – 2010. – Vol. XXIX, N 30. – P. 2985 – 2997.
22. A performance comparison using HPC benchmarks: Windows HPC Server 2008 and Red Hat Enterprise Linux 5 / S. Teige, R. Henschel, H. Li [et al.] // International SPEC Benchmark Workshop. – 2010. – October 8. – 13 p.
23. ANSYS 12.0 Benchmark Performance Data. Windows HPC Server 2008 [Електронний ресурс]. – Режим доступу: http://www.ansys.com/staticassets/ANSYS/staticassets/partner/Windows7/ansys-performance-on-windows.pdf.
24. IBM System/360 Operating System Introduction / IBM Corporation / Fourth Ed. – New York, 1971. – P. 41 – 65.
25. IBM System/360 Operating System MVT Guide / IBM Corporation / Fifth Ed. – New York, 1972. – P. 19 – 25.
26. Goldberg R.P. Architectural principles for virtual computer systems / Goldberg R.P. – Harvard: Harvard University, 1973. – February. – 229 p.
27. Not All Server Virtualization Solutions Are Created Equal [Електронний ресурс] / Andre Metelo // IBM Corporation. – Режим доступу: ftp://ftp.software.ibm.com/software/systemz/Not_All_Server_ Virtualization_Solutions_Are_Created_Equal.pdf.

28. Inside Windows Server 2008 Kernel Changes [Електронний ресурс] / Mark Russinovich // TechNet Magazine. – 2008. – March. – Режим доступу: http://technet.microsoft.com/en-us/magazine/2008.03.kernel.aspx?pr=blog.
29. Intel® 64 and IA-32 Architectures Software Developer's Manual [Електронний ресурс] / Intel Corporation. – Vol. 3C., ch. 23, 25–27. – 119 p. – Режим доступу: http://download.intel.com/design/processor/manuals/253665.pdf.
30. Paravirtualization for HPC Systems / L. Youseff, R. Wolski, B. Gorda [et al.] // International Symposium on Parallel and Distributed Processing and Application (ISPA 2006). – Sorrento, Italy, 2006. – December. – 12 p.
31. Dynamic Virtual Clustering with Xen and Moab / W. Emeneker, D. Jackson, J. Butikofer [et al.] // ISPA06: Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC). – Sorrento, Italy, 2006. – P. 440 – 451.
32. Linux-based virtualization for HPC clusters / L. Nussbaum, F. Anhalt, O. Mornard [et al.] // Proc. of the Linux Symposium. – Montreal, 2009. – July 13–17. – P. 221 – 234.
33. Job execution in virtualized runtime environments in grid / L. Shamardin, A. Demichev, I. Gorbunov [et al.] // 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09). Journal of Physics: Conference Series 219. – Prague, 2010. – 6 p.
34. System-Level Virtualization for High Performance Computing / G. Vallee, T. Naughton, C. Engelmann [et al.] // Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference, (Toulouse, 13–15 Feb. 2008). – Toulouse, 2008. – P. 636 – 643.
35. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors / S. Soltesz, H. Potzl, M. Fiuczynski [et al.] // Proc. of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems. – New York, 2007. – 14 p.
36. Virtual Iron documentation [Електронний ресурс]. – Режим доступу: http://download.oracle.com/technology/products/vm/virtualirondocs/vi_4516_admin.pdf.
37. VMware vSphere 4.1 Documentation [Електронний ресурс]. – Режим доступу: http://www.vmware.com/support/pubs/vs_pages/vsp_pubs_esxi41_i_vc41.html.
38. KVM documentation [Електронний ресурс]. – Режим доступу: http://www.linux-kvm.org/page/Documents.
39. Xen Architecture [Електронний ресурс]. – Режим доступу: http://wiki.xen.org/xenwiki/XenArchitecture.html.
40. Microsoft Hyper-V [Електронний ресурс]. – Режим доступу: http://technet.microsoft.com/en-us/library/cc753637%28WS.10%29.aspx.
41. 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»). – 2010. – Вип. 14. – С. 4 – 6.
42. Hyper-V WMI Provider [Електронний ресурс]. – Режим доступу: http://msdn.microsoft.com/en-us/library/cc136992%28VS.85%29.aspx.
43. Hyper-V: Using Hyper-V and Failover Clustering [Електронний ресурс]. – Режим доступу: http://technet.microsoft.com/en-us/library/cc732181%28WS.10%29.aspx.
44. Task Scheduling Onto Dynamic Large-Scale Parallel Cluster (DLPC) / Hu Kai, Jianwei Niu, Jianping Hu // Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA 2000, (Las Vegas, Nevada, USA, June 24–29, 2000). – Las Vegas: CSREA Press, 2000. – 8 p.
45. Sharifi M. Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques / M. Sharifi, H. Salimi, M. Najafzadeh // The Journal of Supercomputing. – Netherlands: Springer, 2011. – P. 1 – 21.

Стаття надійшла до редакції 06.03.2012