УДК 621.398:007

А. А. ЯРОВИЙ

ПРИКЛАДНІ АСПЕКТИ І ПЕРСПЕКТИВИ ПОБУДОВИ КЛАСТЕРІВ НА ОСНОВІ GPU ДЛЯ РЕАЛІЗАЦІЇ ПАРАЛЕЛЬНОЇ ТА ПАРАЛЕЛЬНО-ІЄРАРХІЧНОЇ ОБРОБКИ ІНФОРМАЦІЇ

Вінницький національний технічний університет,
21021, Хмельницьке шосе, 95, м. Вінниця Україна,
тел.: +380 (432) 580019, E-mail: axa@vinnitsa.com

Анотація. В проведених дослідженнях здійснено аналіз методологічних і прикладних аспектів та перспектив побудови кластерів на базі GPU для реалізації швидкої та ефективної паралельної обробки надвеликих масивів інформації. Наведено результати розробки спеціалізованої програмної бібліотеки для програмування віддалених відеоадаптерів.

Аннотация. В проведенных исследованиях осуществлен анализ методологических и прикладных аспектов и перспектив построения кластеров на базе GPU для реализации быстрой и эффективной параллельной обработки сверхбольших массивов информации. Приведены результаты разработки специализированной программной библиотеки для программирования отдаленных видеоадаптеров.

Abstract. The analysis of methodological and applied aspects and prospects of construction clusters on base of GPU for realization of fast and effective parallel processing of the superlarge arrays of the information is carried out in the given researches. The results of development of the specialized program library for programming the remote videoadapters are given.

Ключові слова: паралельні обчислення, кластери, GPGPU, програмування відеоадаптерів, паралельно-ієрархічне перетворення.

ВСТУП

Останнім часом в наукових колах все ширше обговорюється тема високопродуктивних обчислень і суперкомп'ютерів. Вони непогано освоєні, для них розроблено велику кількість прикладних пакетів, з їх допомогою вирішуються складні обчислювальні задачі. Вони мають багато переваг, за винятком однієї – високої вартості, що змушує шукати альтернативу. Одним з таких способів є застосування графічних процесорних пристроїв (Graphical Processor Unit – GPU) для створення систем, здатних гідно конкурувати з традиційними суперкомп'ютерами [1-4].

У минулих роботах було доведено, що досить вдалим розв’язком проблеми швидкої та ефективної паралельної обробки надвеликих масивів даних на основі спеціалізованих системних рішень, зокрема нейроподібних паралельно-ієрархічних систем, є використання відеоадаптерів для обчислень загального призначення (GPGPU) [4-7]. Необхідно відзначити, що найбільшої ефективності обчислень на GPU вдається досягти при виконанні матричних обчислень. У контексті GPU під матричними обчисленнями розуміється множення застосовні лише до матриць над кільцями. При теперішніх обсягах пам'яті на GPU можна перемножувати матриці розмірністю до 8000x8000, і всі дані будуть розміщені у відеопам'яті. Реально досягнута продуктивність також досить висока: на AMD HD 2900 – 100 GFLOPS, на nVidia GeForce GTX8800 – 125 GFLOPS (на процедурі з CUBLAS, реалізованою nVidia). Така продуктивність досягається за рахунок використання блокових алгоритмів множення матриць, а класичні реалізації виявляються на порядок повільнішими [1,2].

Як окремий специфічний приклад необхідно навести задачу розрахунку ціни опціонів по формулі Блека-Шоулза, що часто вирішується на практиці. Це той окремий випадок, коли велика кількість обчислень поєднується з дуже простою паралельною структурою. На вхід подаються три масиви параметрів опціонів, і потрібно розрахувати вартість опціону для кожної трійки параметрів. Формула для розрахунків містить лише одне розгалуження, а решою – абсолютно лінійна. Більше того, в розрахунках активно використовуються операції логарифма та експоненти, які набагато ефективніше реалізовані на сучасних графічних процесорах (GPU), ніж на центральних процесорах (CPU). В результаті системи на GPU демонструють на даній задачі практично 100-кратне прискорення в

порівнянні із звичайним процесором. На nVidia GeForce 8800GTX досягається продуктивність, що складає 260 GFLOPS [1-3].

Ще одне завдання – обробка зображень – була однією із перших задач, що вирішуються на GPU. Найбільш важливими алгоритмами тут є фільтрація зображень і перетворення Фур'є. Фільтрація, завдяки відносно простий структурі, легко відображується на GPU, проте коефіцієнт повторного використання даних значно менший, ніж у множення матриць, що і пояснює значно меншу ефективність GPU при вирішенні даної задачі. При розмірі ядра 3x3 продуктивність коду на nVidia GeForce 8800GTX складає близько 14,1 GFLOPS (тобто завантаження – всього 4%). Часто вживана на практиці окрема фільтрація за допомогою фільтра Гауса показує гірші результати, зважаючи на ще нижчий коефіцієнт повторного використання.

Перетворення Фур'є відноситься до класу алгоритмів, які на GPU виконуються за час, пропорційний NlgN, де N – розмір завдання. Обидва алгоритми реалізуються за допомогою багатоетапної схеми, де на кожному етапі („проході") застосовується ядро типу „метелика". І хоча обчислювальна складність подібного ядра відносно невисока, GPU від nVidia вдається досягти на ньому непоганих результатів. На перших етапах роботи весь робочий набір поміщається у статичну пам'ять, що дозволяє зібрати його в один етап і отримати порядка 40Gflops на графічному адаптері nVidia GeForce 8800GTX [1,2].

В той же час існує багато завдань, які дуже добре масштабуються і на кластерах GPU, і навіть в розподіленому обчислювальному середовищі, побудованому із графічних процесорів. Кластери з GPU дозволяють отримати високу пікову продуктивність, проте наблизитися до неї на практиці непросто. Найбільш істотним фізичним обмеженням є низька пропускна спроможність каналу „GPU – центральний процесор (CPU)", особливо у напрямку до центрального процесора, оскільки мережева карта безпосередньо з GPU недоступна. З поширенням PCI-Express та швидкого зворотного пересилання ситуація значно покращилась: швидкості порядку 1 Гбайт/с зробили кластери GPU ефективними і реально використовуваними комп'ютерними системами.

МЕТА ДОСЛІДЖЕННЯ

Метою роботи є дослідження методологічних і прикладних аспектів та перспектив побудови кластерів на базі GPU для реалізації швидкої та ефективної паралельної обробки надвеликих масивів інформації на основі спеціалізованих системних рішень, зокрема нейроподібних паралельно-ієрархічних систем, а також реалізація спеціалізованої програмної бібліотеки для програмування віддалених відеоадаптерів.

ПОСТАНОВКА ПРОБЛЕМИ. АНАЛІЗ ЕВОЛЮЦІЇ ТА СТРУКТУРНО-ФУНКЦІОНАЛЬНОЇ ОРГАНІЗАЦІЇ GPGPU

Спочатку графічні процесорні пристрої не призначалися для високопродуктивних обчислень. Завданням GPU першого покоління, що, завдяки комп'ютерним іграм, набули широкого поширення в середині 90-х років, була побудова в реальному часі зображень за описом тривимірних сцен. Для цього була потрібна швидка однорідна обробка великої кількості елементів (вершини, трикутники, пікселі), яка досягалася, перш за все, за рахунок апаратної реалізації основних алгоритмів.

Перші можливості програмування з'явилися в 2001 році в GPU другого покоління. Програми, що виконуються на GPU, почали називатися «шейдерами» (від англ. shade – «зафарбовувати»), і основним їх призначенням було обчислення кольору пікселя на екрані. Тоді ж почали здійснюватися перші спроби використання GPU для спільних обчислень, проте через низьку точність обчислень, що складала лише 16 біт у форматі з „фіксованою комою", і обмежених можливостей програмування, – довжина шейдера не перевищувала 20 команд – це не набуло поширення. Проте, ситуація змінилася з появою в 2003 році серії nVidia GeForce FX, що надала підтримку повноцінній 32-розрядній арифметиці дійсних чисел. Тоді ж був запропонований термін „спільні обчислення на GPU" (General Purpose GPU – GPGPU). Графічні процесори почали активно застосовуватися для вирішення завдань математичної фізики, обробки зображень і матричних обчислень [1,2].

GPU першого покоління – GeForce 6, GeForce 7, ATI Radeon X1K і пізніші – ще більше розширили можливості програмування за рахунок введення додаткових команд – управління, зокрема підтримка повноцінних циклів.

Сучасні GPU третього покоління, такі як nVidia GeForce 8, nVidia GeForce 9, AMD HD 2K і AMD HD 3K, містять набір однакових обчислювальних пристроїв – потокових процесорів (ПП), що працюють із спільною пам'яттю GPU (відеопам'ять). Число ПП міняється від чотирьох до 128, розмір відеопам'яті може досягати 1 Гбайт. Всі ПП синхронно виконують один і той же шейдер, що дозволяє віднести GPU

до класу SIMD (Single Instruction, Multiple Data). За один прохід, що є етапом обчислень на GPU, шейдер виконується для всіх точок двовимірного масиву. Система команд ПП включає арифметичні команди для дійсних чисел та цілочисельних обчислень з 32-розрядною точністю, команди управління (розгалуження і цикли), а також команди звернення до пам'яті. GPU виконують операції лише з даними на регістрах, число яких може досягати 128. Через високі затримки (до 500 тактів) команди доступу до оперативної пам'яті виконуються асинхронно. З метою приховування затримок в черзі виконання GPU може одночасно знаходитися до 512 потоків, і якщо поточний потік блокується за доступом до пам'яті, на виконання ставиться наступний. Оскільки контекст потоку повністю зберігається на регістрах GPU, перемикання здійснюється за один такт – за цю операцію відповідає диспетчер потоків, який не є програмованим [1-3, 7].

Тактові частоти GPU нижчі, ніж у звичайних процесорів, і лежать в діапазоні від 0,5 до 1,5 ГГц, проте завдяки великій кількості потокових процесорів продуктивність GPU досить значна. Сучасні GPU верхнього цінового сегменту мають пікову продуктивність 200-500 GFLOPS, що у поєднанні з можливістю установки в одну машину двох графічних адаптерів дозволяє отримати пікову продуктивність в 1 TFLOPS на одному персональному комп'ютері. Більш того, на деяких реальних задачах досягається до 70% пікової продуктивності. Одночасно з цим, порівняно з класичними кластерними системами, GPU володіють значно кращими характеристиками як за ціною (менше 1 дол. на GFLOPS), так і за енергоспоживанням (менше 1 Вт на GFLOPS).

На підтвердження вищенаведеного необхідно відзначити, що в 2007 році компанія nVidia почала виробництво апаратних рішень на основі GPU, спеціально орієнтованих на виконання обчислювальних задач. Ця серія називається „Tesla" і включає конфігурації на основі одного, двох і чотирьох графічних адаптерів. Основний орієнтир використання Tesla – серверні системи [1-3, 7].

Архітектура GPU сімейства nVidia GeForce 8 представлена на рис. 1. Взагалі кажучи, вона не є класичним представником архітектури SIMD. У ній 128 ПП об'єднані в SIMD-групи по 16 мультипроцесорів (МП), але при цьому різні МП працюють незалежно один від одного, хоча і виконують один і той же шейдер. Кожен ПП є суперскалярним пристроєм і може виконувати до двох команд за такт. При зверненні у відеопам'ять йому доступна вся пам'ять, як на читання, так і на запис [1-3, 7]. Проте на практиці зважаючи на слабкість засобів синхронізації між різними МП бажано процес обробки будувати так, щоб адреси запису не перетиналися.

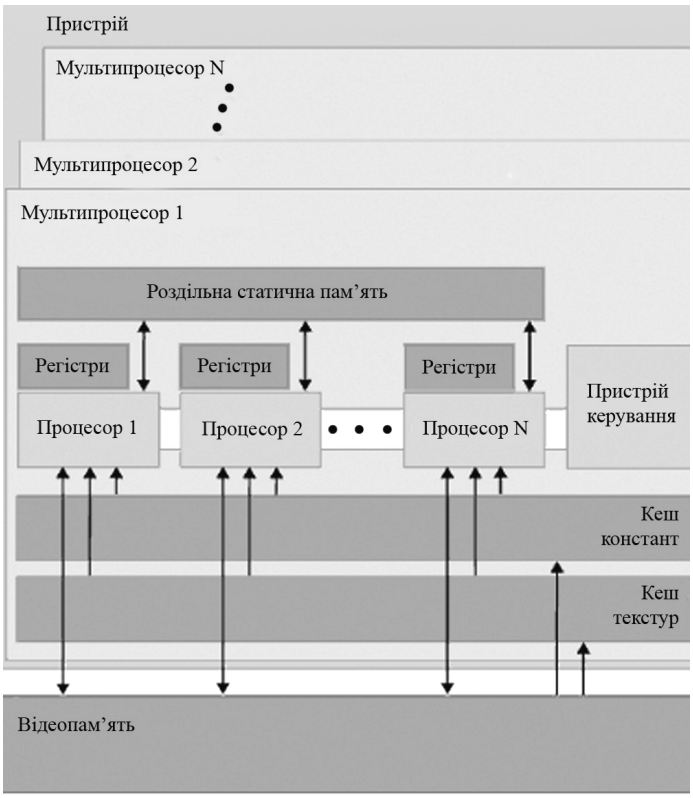

Рис. 1. Архітектура GPU сімейства Nvidia GeForce 8 [1]

У GeForce 8 програмістові доступні 16 Кбайт статичної пам'яті на мультипроцесорі, що явно адресується, яка використовується спільно всіма ПП одного мультипроцесора, що має всередині механізм бар'єрної синхронізації потоків. Відзначимо, що грамотне використання статичної пам'яті якраз і є ключем до написання ефективних програм для даної архітектури.

На відміну від nVidia, потоковий процесор (ПП) AMD має архітектуру з довгим командним словом. Один ПП містить шість функціональних пристроїв (ФП), з яких один відповідає за команди переходу, чотири – за звичайні арифметичні команди і ще один – за обчислення елементарних функцій (log, sin і т.і.). Арифметичні ФП здатні виконувати до двох команд за такт, а ФП, що обчислює елементарні функції, – одну. Таким чином, один такий ПП здатний виконувати до восьми команд за такт, що більше, ніж у nVidia.

При розгляді даної архітектури можна в першому наближенні обмежитися тільки командами, що задають одну і ту ж операцію для чотирьох арифметичних ФП. Це тим більше логічно, що обмін даними з пам'яттю здійснюється в термінах 128-бітових одиниць, а це дорівнює чотирьом 32-розрядним словам. З цієї точки зору ПП від AMD схожий на один елемент SPE (Synergistic Processing Element) процесора Cell або ядро x86-процессора з розширеннями SSE2 [1-3, 7].

Доступ до пам'яті на читання можливий через один з 32 сегментних регістрів, кожен з яких адресує двовимірну ділянку пам'яті, що містить до 8192*8192 елементів (один елемент = 128 біт). На відміну від nVidia, запропоновані компанією “AMD” GPU надають два режими запису. У режимі регулярного запису шейдер може записувати по одному елементу в кожен з вихідних буферів, при цьому індекс елементу в буфері жорстко заданий. У режимі довільного запису доступний тільки один буфер, зате запис може проводитися за довільною адресою. Засоби синхронізації відсутні.

Серія GPU від AMD, що орієнтована на високопродуктивні обчислення, має назву FireStream. Відмітимо, що в FireStream 9170 вперше з'явилася підтримка обчислень з подвійною точністю, при цьому пікова продуктивність складає 100 GFLOPS [1-3, 7].

Перед подальшим розглядом процесу побудови кластерів на основі GPU, необхідно зазначити, що, цілком природно, далеко не на всіх задачах вдається отримати високу продуктивність навіть на одному GPU. Тут можна виділити спільні системні вимоги для задач, що потребують реалізацію обчислень загального призначення на GPU:

1. Висока обчислювальна потужність. Обсяг обчислень повинен значно перевищувати обсяг необхідних даних, інакше обчислення на GPU не покриють навіть накладних витрат на пересилання вхідних даних і результатів.

2. Великий розмір обчислюваного масиву. Запуск проходу (етапу) – досить дорога операція, більш того, при малій розмірності масиву не буде достатньої кількості потоків для покриття витрат на читання з пам'яті.

3. Можливість незалежного обчислення цільових елементів на кожному з проходів. Ця вимога безпосередньо є наслідком відсутності можливостей синхронізації.

4. Масиви даних, використовувані при обчисленні, повинні цілком поміщатися у відеопам'ять. Це є незаперечним наслідком повільного обміну між оперативною та відеопам'яттю [1,2].

Зважаючи на наведені системні вимоги можна стверджувати, що організація паралельно-ієрархічної обробки інформації є тією задачею, що входить до класу задач, які ефективно реалізовувати на GPU-платформах [8].

ПРИКЛАДНІ АСПЕКТИ ПОБУДОВИ КЛАСТЕРІВ НА ОСНОВІ GPU. РОЗРОБКА СПЕЦІАЛІЗОВАНОЇ ПРОГРАМНОЇ БІБЛІОТЕКИ ДЛЯ ПРОГРАМУВАННЯ ВІДДАЛЕНИХ ВІДЕОАДАПТЕРІВ.

Насамперед, необхідно відзначити, що наукові дослідження з розробки та реалізації кластерних систем на основі GPU є не новинкою, а є загальновизнаними, мають свою історію та практичні результати.

Зокрема, ще у 2008 році на конференції „International Supercomputing” компанія Acceleware, виробник HP-систем, представила перший у світі комерційно доступний кластер на базі GPU – C30-16. Таке проектне рішення було комбінацією 64 потужних GPU і технологій кластеризації компанії Acceleware з метою забезпечення високої продуктивності та масштабованості для великих замовників. У ролі останніх виступали компанії, зайняті розвідкою нафтових родовищ, геологи, яким потрібні потужні обчислювальні ресурси. Апаратна частина C30-16 містила один головний та 16 робочих вузлів, взаємодія між якими забезпечувалась за рахунок DDR Infiniband, 24-портового комутатора Mellanox DDR і мережі Gigabit Ethernet (2 з'єднання на вузол) [9].

В березні 2009 року компанія Penguin оголосила про випуск кластера Penguin Altus 1702 на базі

Tesla GPU. Нове рішення має на меті зробити доступними високопродуктивні обчислення для наукових і інженерних потреб. Серверний кластер Altus 1702 поєднує чотири обчислювальні системи NVIDIA Tesla S1070 1U GPU (кожна на чотирьох процесорах Tesla T10 GPU), Gigabit Ethernet та керуюче ПЗ Penguin Scyld. Віртуалізацію кластерного середовища забезпечує Scyld ClusterWare, що має єдину точку керування – майстер-вузол, за допомогою якого можна швидко зібрати з наявних ресурсів необхідну конфігурацію. У конфігурації Z170034 обчислювальна потужність запропонованої системи досягає 16 Tflops. Також випускаються конфігурації з вісьма Tesla S1070 GPU продуктивністю понад 32 Tflops. Всі системи працюють на процесорах AMD Opteron 2376, оснащені 8ГБ оперативної пам'яті на вузол і 24-портовим комутатором GigE [10].

В травні 2009 року компанія NVIDIA та її партнери повідомили про доступність кластера на базі Tesla™ GPU – повністю готового до роботи рішення, що дозволяє науковцям та IT-менеджерам з легкістю додавати обчислювальні можливості GPU в існуючі центри обробки даних. Кластери на базі Tesla GPU забезпечують продуктивність до 30 разів вище в порівнянні з рішеннями на CPU. Такі кластери споживають значно менше енергії, що спричиняє скорочення витрат, при цьому вони дозволяють вирішувати найбільш вимогливі до ресурсів завдання в таких галузях, як гідродинаміка, молекулярна динаміка, обробка сейсмічних даних і фінансовий аналіз. Найбільший французький банк BNP Paribas недавно замінив у відділенні по корпоративному банкінгу та інвестиціям 500 традиційних CPU, що споживають 25 кВт, на невеликий кластер, що складається з CPU серверів і двох систем Tesla S1070 1U, що споживають всього 2 кВт. Крім значного прискорення операцій, досягнутого за допомогою Tesla GPU, більше економічний кластер дозволив BNP Paribas скоротити споживання енергії в 190 разів. Кластери з Tesla складаються з x86 CPU серверів із системами Tesla S1070 1U GPU. Конфігурації мають мінімальну продуктивність в 16 Tflops, забезпечувану чотирма Tesla S1070, кожний з яких містить чотири GPU серії Tesla 10. Всі системи містять хост-сервери, infiniband-перемикачі та кабелі й мають широкі можливості для настроювання під певні вимоги користувачів [11].

Таким чином, компактне виконання з високою обчислювальною щільністю поряд з високошвидкісними каналами передачі даних дозволяють використати проектні рішення на базі GPU у компактних кластерних комплексах.

На рис. 2 представлено гібридний кластер із застосуванням GPU як основних обчислювачів. Такий гібридний кластер розрахований на пікове навантаження близько 8 Tflops. Кожний вузол GPU виконаний у конструктиві 1U та має 4 незалежних GPU обчислювача, 16Gb оперативної пам'яті й два швидкісних канали для підключення із сервера керування вузлом. Сервери керування виконують роль кешуючих пристроїв та являють собою двопроцесорні багатоядерні системи, обладнані двома контролерами PCI Express x16 G2 для забезпечення високої пропускної здатності комплексу. Загальне керування роботою кластера й розподіл завантаження ресурсів покладено на керуючий сервер. Мережі передачі даних і керування побудовані з використанням протоколу TCP на базі Gigabit Ethernet [12].

Рис. 2. Структурна організація гібридного кластера на основі GPU, де ДБЖ – джерело безперебійного живлення [12]

Отже, в проведених дослідженнях обчислювальний кластер з використанням програмованих відеоадаптерів успадкує усі особливості обох підходів, але допоможе подолати існуюче апаратне

обмеження на кількість відеоадаптерів у одній обчислювальній системі. Таким чином, означена обчислювальна платформа у найпростішому випадку має складатися з деякої кількості машин виконавців та машини координатора. Такий апаратно-програмний комплекс є найбільш доступним, його простіше програмувати. Кожна машина виконавець (далі виконавець) має містити хоча б один відеоадаптер, а завдання для кожного виконавця бути паралельною відносно даних програмою, яку можливо виконати на відеоадаптері (шейдером). При цьому програма для кластера, має керувати виконанням кожного завдання на окремому виконавці, оскільки інакше кластерне програмне забезпечення мало б керувати відправкою даних з відеоадаптера одного виконавця, до відеоадаптера іншого, що неможливо, оскільки шейдер не може звернутися до програми що виконується на CPU з запитом про такі дані (та й взагалі з будь-яким запитом).

Основа логічної структури системи витікає з поставленої задачі. Тобто на машині-координаторі має бути запущено програму, яка б дозволила відсилати машинам-виконавцям завдання на виконання. Кожне завдання – це шейдер та дані для нього. У найпростішому випадку усі виконавці отримують однакові шейдери, це добре узгоджується з паралельною відносно даних моделлю виконання шейдерів. Розроблена програмна бібліотека спрощує керування виконанням обчислень на множині виконавців індивідуальних завдань, тобто автоматизує процес написання програми-координатора. Якщо використовуються однакові за потужністю відеоадаптери – кластер гомогенний, якщо різні – гетерогенний. (певна умова однорідності зберігається, оскільки розроблена бібліотека базується на ATI Stream SDK).

Розроблена бібліотека виконана у вигляді ASP.NET сервісу який після інсталяції на IIS Server дозволяє за допомогою proxy об’єкту використовувати функціональність бібліотеки useGPU (діаграма класів наведена на рис. 3) користувацькій програмі з віддаленого комп’ютеру, тобто приховує від користувача більшу частину роботи з низкорівневими об’єктами ATI StreamComputing SDK, такими як ідентифікатор ресурсу, ідентифікатор області пам’яті (як можна побачити з діаграми класів ці об’єкти інкапсульовано у структурі Mem), інформація та атрибути відеоадаптера, його унікальний ідентифікатор (містяться у структурі Device), контекст виконання, використовуючу розроблену нами раніше програмну бібліотеку useGPU.

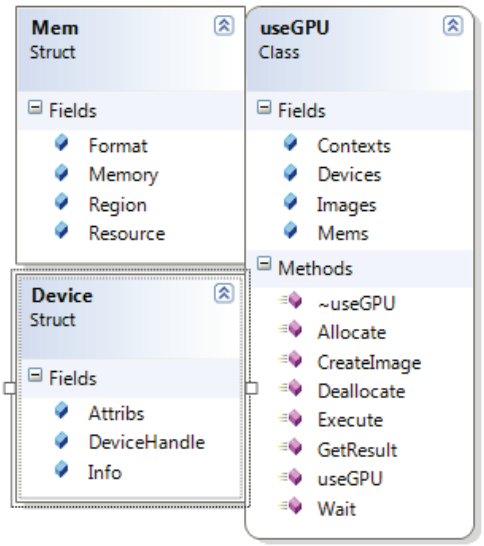

Рис. 3. Класова діаграма використаної програмної бібліотеки „useGPU”

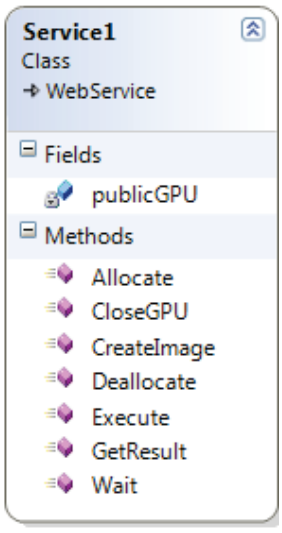

Рис.4. Класова діаграма розробленої програмної бібліотеки „dotGPU”

Усі програми в межах розробленої програмної бібліотеки „dotGPU”, що використовуватимуться для обчислень відеоадаптера мають містити такі основні етапи:

- ініціалізація об’єкту Service1SoapClient, що інкапсулює функціональність зв’язку з API ATI StreamComputing SDK;
- завантаження параметрів програми, що буде виконано на відеоадаптері (далі шейдеру) , за допомогою методу Allocate;
- створення образу шейдеру, за допомогою методу CreateImage;
- виконання створеного образу шейдеру над завантаженими параметрами, з використанням методу Execute;

- очікування закінчення виконання програми (перевірка чи не закінчилася обробка даних виконується за допомогою методу Wait);
- завантаження результатів обчислення в основну пам'ять за допомогою методу GetResult;
- якщо потрібно звільнити пам'ять відеокарти для нових даних, то треба виклакати метод Deallocate;
- після закінчення обчислень та отримання результатів, якщо більше не потрібно використовувати відеоадаптер слід викликати метод closeGPU.

Для програмної реалізації було обрано платформу „.NET framework", оскільки ця платформа дозволяє пришвидшити розробку програмного забезпечення (відносно „C++"). Зрозуміло, що програма написана для .NET framework потребуватиме у 3-5 разів більше ресурсів CPU, але оскільки у нашому програмно-апаратному комплексі використовуватиметься тільки обчислювальна потужність відеоадаптера – ця обставина не матиме впливу на швидкодію комплексу.

Обмеження розробленої програмної бібліотеки є результатом вибору технологій реалізації. ATI Stream SDK дозволяє програмувати тільки відеоадаптери AMD/ATI [4-7]. Використання ASP.NET Services обмежує розміри масивів, що використовуються як параметри шейдерів. Ці обмеження наслідок того, що ASP.NET Services реалізовано на основі специфікацій SOAP.

Обмеження на топологію кластерної мережі визначаються із гранично можливих апаратних ресурсів, оскільки реалізація мережевої топології – повністю завдання користувача. У якості операційної системи можна використовувати лише Windows XP та новіші, з встановленим .NET framework та IIS 6+. І хоча платформа mono.net теж підтримує необхідну функціональність ASP.NET Services, але mono runtime не може виконувати програми в який MS IL перемежовується з ассемблером ЦП (змішані збірки).

Для чіткішого означення перспектив розвитку запропонованої кластерної структури на основі GPU в подальших дослідженнях буде проведено багатоетапне тестування програмної бібліотеки на справжньому кластері машин з відеоадаптерами ATI. Зрозуміло, що найцікавішими та найважливішими тут є питання організації керування розподіленням задач у гетерогенному кластері та питання обмежень мережевого протоколу, що використовується.

ВИСНОВКИ

Таким чином, на сьогодні є хороша перспектива використання графічних процесорів для високопродуктивних обчислень. Для розглянутих архітектур GPU чітко простежуються як розрив між високою піковою продуктивністю і можливістю її реального досягнення для широкого класу завдань, так і відсутність адекватних засобів програмування. Проте якщо прослідкувати вектор розвитку за першими трьома поколіннями, то можна відзначити, що вони змінюються у бік збільшення їх універсальності при подальшому зростанні продуктивності. Важливим кроком в цьому напрямку є GeForce 8, який відходить від традиційної SIMD-архітектури для GPU. У тому ж напрямку рухаються не лише GPU. Дійсно, процесор Cell, експериментальні 80-ядерні процесори Intel та графічний процесор Larrabee від Intel, що знаходиться в даний час на стадії розробки, також перебувають в руслі цих тенденцій. Спостерігається конвергенція різних паралельних архітектур у бік створення високопродуктивної і доступної MIMD-архітектури загального вигляду. Паралелізм в сучасній архітектурі не обмежується суперкомп'ютерами – він поширюється на завдання обчислень загального призначення у вигляді багатоядерних процесорів, GPU, ПЛІС та інших архітектур [1-3, 7].

В ході проведених наукових досліджень було проаналізовано методологічні і прикладні аспекти та перспективи побудови кластерів на базі GPU для реалізації швидкої та ефективної паралельної обробки надвеликих масивів інформації на основі спеціалізованих системних рішень, зокрема нейроподібних паралельно-ієрархічних систем. Також в дослідженнях, на рівні детального аналізу структурної організації та програмно-апаратних особливостей організації паралельних обчислень в графічних апаратних пристроях, визначено системні вимоги для задач, що потребують реалізації обчислень загального призначення на GPU.

На основі наведеного аналізу було поставлено та вирішено задачу розробки спеціалізованої програмної бібліотеки для програмування віддалених відеоадаптерів. Окреслено шляхи можливого подальшого вдосконалення даної програмної бібліотеки.

СПИСОК ЛІТЕРАТУРИ

1. Графический вызов суперкомпьютерам [Електронний ресурс] / А. Адинец, Вл. Воеводин // Открытые системы – 2008. – №4 – Режим доступу: http://www.osp.ru/os/2008/04/5114497.
2. GPU: эволюция / Информационно-аналитический портал Overclockers.com.ua [Електронний ресурс]

– Режим доступу: http://www.overclockers.com.ua/video/gpu-evolution.
3. Сравнение производительности графических ускорителей и центрального процессора при вычислениях для больших объемов обрабатываемых данных / Скрибцов П.В., Долгополов А.В. // Нейрокомпьютеры: разработка, применение – Москва, Издательство "Радиотехника", 2007. – № 9. – С. 421-425.
4. Прикладна реалізація масштабних нейронних та нейроподібних паралельно-ієрархічних мереж на основі технологій GPGPU [Електронний ресурс] : [Електронне наукове фахове видання] / А. А. Яровий, Ю. С. Богомолов, К. Ю. Вознесенський // Наукові праці Вінницького національного технічного університету. – 2009. – №2. – С. 1-8. – Режим доступу до журн.: http://www.nbuv.gov.ua/e-journals/VNTU/2009_2_ua/2009-2.files/uk/09aayogt_ua.pdf.
5. Імітаційне моделювання та програмна емуляція нейроподібних паралельно-ієрархічних систем на основі технологій GPGPU. / А.А. Яровий, Ю. С. Богомолов. : матеріали IV Міжнародної науково-технічної конференції [Сучасні проблеми радіоелектроніки, телекомунікацій та приладобудування “СПРТП-2009”], (Вінниця, 8-10 жовтня 2009 р.). Частина 2. – Вінниця, ПП „Радіоінформ”, 2009 – С. 27.
6. Аналіз функціонування програмної моделі GPGPU в контексті організації паралельних обчислень в нейроподібних паралельно-ієрархічних системах / А.А. Яровий // Оптико-електронні інформаційно-енергетичні технології. – 2009. – №1 (17). – С. 42-49.
7. Методологічні особливості реалізації нейроподібних паралельно-ієрархічних систем на основі технологій GPGPU / Кожем'яко В.П., Яровий А.А., Богомолов Ю.С., Вознесенський К.Ю. // Оптико-електронні інформаційно-енергетичні технології. – 2008. – №2 (16). – С. 26-33.
8. Прикладные аспекты программно-аппаратной реализации нейроподобных параллельно-иерархических систем / Яровой А.А. : Сборник научных трудов. [Научная сессия МИФИ – 2009], [XI Всероссийская научно-техническая конференция «Нейроинформатика-2009»], (Москва, 27-30 января 2009 г.), [В 2 частях. Ч. 2.] – Москва, МИФИ, 2009. – С. 39-48.
9. Acceleware выпустила первый в мире коммерчески доступный кластер на базе GPU NVIDIA [Електронний ресурс] // Журнал iXBT.com. – 2009. - №10. - Режим доступу: http://www.ixbt.com/news/all/index.shtml?10/63/33.
10. Penguin выпустила кластер на NVIDIA Tesla GPU [Електронний ресурс] // Компьютерное обозрение. – 2009. - Режим доступу: http://ko-online.com.ua/node/41548.
11. Кластер на базе Tesla ускоряет научные исследования [Електронний ресурс] // Официальный сайт NVIDIA Corporation. – Режим доступу: http://www.nvidia.ru/object/io_1241767854048.html.
12. Персональный суперкомпьютер ARBYTE SC [Електронний ресурс] // Официальный сайт ARBYTE Computers. – Режим доступу: http://www.arbyte.ru/products/HPC/supercomputer.shtml.

Надійшла до редакції 06.05.2009 р.

ЯРОВИЙ А.А. – к.т.н., доцент, доцент кафедри інтелектуальних систем, науковий співробітник кафедри лазерної і оптоелектронної техніки, Вінницький національний технічний університет, Вінниця, Україна.