УДК 004.424

А.Ю. Дорошенко, І.В. Оконський, К.А. Жереб, О.Г. Бекетов

ВИКОРИСТАННЯ ЗАСОБІВ МОДЕЛЮВАННЯ ДЛЯ ВИЗНАЧЕННЯ ОПТИМАЛЬНИХ ПАРАМЕТРІВ ВИКОНАННЯ ПРОГРАМ НА ВІДЕОГРАФІЧНИХ ПРИСКОРЮВАЧАХ

Побудовано інструментарій моделювання гетерогенних Грід-систем з відеографічними прискорювачами gpusim, який використано для моделювання часу виконання гравітаційної задачі взаємодії N тіл, що залежить від параметру. Засоби моделювання дозволяють обрати найбільш оптимальне значення параметру в автоматизованому режимі і при цьому суттєво скоротити час підбору параметрів у порівнянні з безпосереднім виконанням програми.

Вступ

У роботі [1] автори запропонували систему моделювання виконання задач на відеографічних прискорювачах (graphical processing units, GPU). Запропонована система gpusim побудована на основі платформи GridSim [2], і дозволяє моделювати виконання задач на гетерогенних системах, що складаються зі звичайних процесорів (central processing units, CPU) та GPU.

Дана робота продовжує розпочаті в [1] дослідження. Основний внесок роботи полягає в наступному. По-перше, розглядається більш складна прикладна задача, а саме задача гравітаційної взаємодії N тіл [3]. По-друге, ставиться більш складне питання, а саме пошуку оптимального значення параметра для отримання більш ефективної програми.

У роботі проведено дослідження залежності часу виконання програми на відеографічному прискорювачі на прикладі задачі N тіл від заданого параметра (кількість потоків на блок). Отримано оптимальні значення для даного параметра. Побудовано модель часу виконання для задачі N тіл, що дозволяє отримувати оптимальне значення параметра без запуску обчислень. Це дозволяє суттєво скоротити час розробки оптимальної програми. Зокрема, для N = 2^21 оптимальне значення параметра 256 отримано методом моделювання за 3 хвилини, тоді як його підтвердження експериментальним шляхом потребувало 8 годин обчислень.

Задача N тіл широко розглядається у літературі, зокрема, останнім часом вона часто розв'язується на GPU. Ця задача може використовуватись як в астрофізиці для опису еволюції космічних об'єктів [4 – 7], так і в молекулярній динаміці для моделювання структури речовини [7 – 8]. Відмінність даної роботи полягає у використанні засобів моделювання для знаходження оптимальних значень параметрів виконання програми.

Матеріал даної роботи організований наступним чином. У розділі 1 наведено постановку задачі N тіл і досліджено залежність часу виконання паралельного GPU-алгоритму від параметрів. Розділ 2 присвячено побудові наближеної моделі часу виконання задачі N тіл у рамках інструментарію gpusim. У розділі 3 наведено дослідження побудованої моделі та підтверджено її адекватність. Роботу завершують висновки та напрямки подальшої роботи.

1. Постановка задачі

Розглядається класична задача ньютонівської механіки еволюції системи N тіл відомої маси в тривимірному евклідовому просторі, що попарно взаємодіють під дією гравітаційних сил [3]. В початковий момент часу задані положення тіл в просторі та їх швидкості. Мета – прогнозувати стан системи в наступні моменти часу. Поведінка системи описується задачею Коші з $2N$ диференціальних рівнянь

першого порядку. Задача не є розв'язною аналітично в загальному випадку, тому для її чисельного інтегрування використано наближений чисельний метод дискретизації за часовим параметром типу предиктор-коректор з використанням інтерполяційних многочленів Ерміта [9].

Реалізовано паралельний алгоритм обчислення цієї задачі на GPU. На рис. 1 показано графік залежності часу виконання даного алгоритму в залежності від розміру задачі (кількості тіл) N на графічному прискорювачі Nvidia GeForce GTX 650 (384 CUDA Cores, 1058 MHz, 86,4 GFLOPS FP64, 1024 MB) [10], який використовувався для отримання експериментальних даних у подальшій роботі. Прогнозований час виконання одного кроку алгоритму для зазначеного прискорювача становить

$$T_{GPU} = \frac{N^2}{8 \cdot 10^8}$$

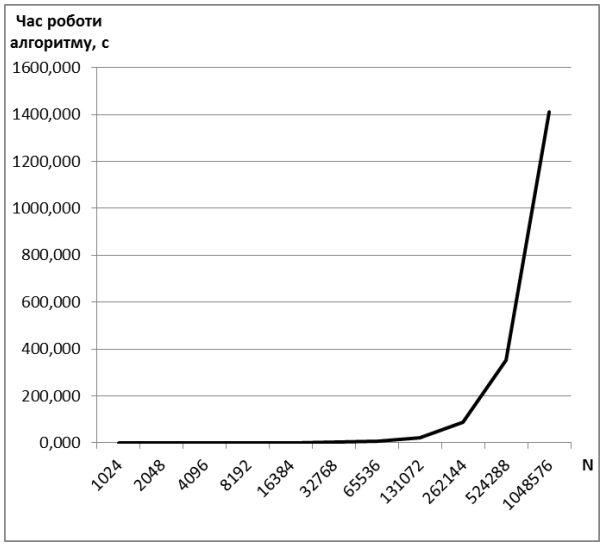

Рис. 1. Залежність часу виконання алгоритму в секундах від N

Оскільки залежність часу виконання від розміру задачі має квадратичний характер, для великих розмірів задачі актуальним є питання оптимізації паралельного алгоритму, зокрема шляхом підбору параметрів виконання.

Одним з параметрів, що суттєво впливають на час виконання програми на GPU, є розбиття задачі на потоки та блоки потоків. Зокрема, параметром, яким може керувати розробник, є кількість потоків на блок (threads per block, TPB). Згідно інструкції користувача CUDA v.5.0 мінімальне та максимальне значення цього параметра складають 1 та 1024 відповідно. Для вирішення прикладних задач рекомендовано використовувати 256 потоків на блок і задавати це значення статично в коді програми [11]. Проте існує можливість оптимального вибору цього параметра, що збільшує ефективність програми.

Початковий алгоритм був доопрацьований для можливості програмного задання будь-якого значення кількості потоків на блок у діапазоні обмежень CUDA – [1; 1024]. Внесені доопрацювання не вплинули на характеристики алгоритму, що було перевірено порівнянням результатів роботи та часу виконання на однакових вхідних даних оригінальної та доопрацьованої версій. У подальших дослідженнях використовується доопрацьована версія алгоритму, для якої значення кількості потоків на блок також є вхідним параметром.

Для дослідження впливу значення кількості потоків на блок на час виконання алгоритму було проведено серію експериментів для N = [2^10; 2^20] та TPB = = [1; 1024], із мультиплікативним кроком 2. Результати експериментів показані на рис. 2 – 4. Згідно отриманих результатів, для різних значень N мінімальний час виконання алгоритму досягається при різних значеннях TPB: для малих N оптимальне значення TPB менше, ніж для великих.

Подальше дослідження паралельного алгоритму обчислення гравітаційної задачі взаємодії N тіл передбачає створення наближеної моделі для розробленого у [1] середовища моделювання гетерогенних паралельних систем gpusim, оскільки для проведення експериментів для N, більших за 2^20, потребується багато часу: вже для N = 2^21 експеримент триває 8 годин. Інтерес становить залежність часу виконання від N та TPB, таким чином, вхідними даними для моделі є діапазони вхідних значень N та TPB, а вихідними – час виконання алгоритму на паралельній системі.

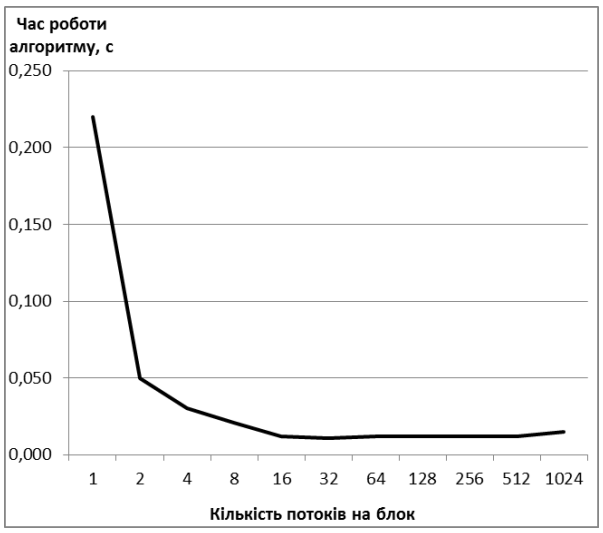

Рис. 2. Залежність часу виконання алгоритму від TPB при N = 2^10

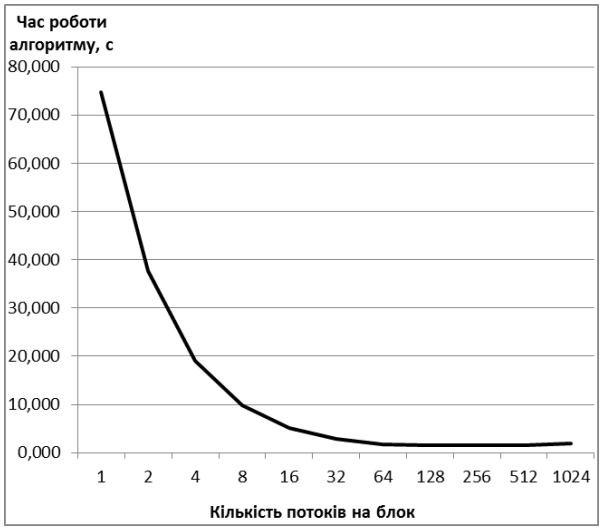

Рис. 3. Залежність часу виконання алгоритму від TPB при N = 2^15

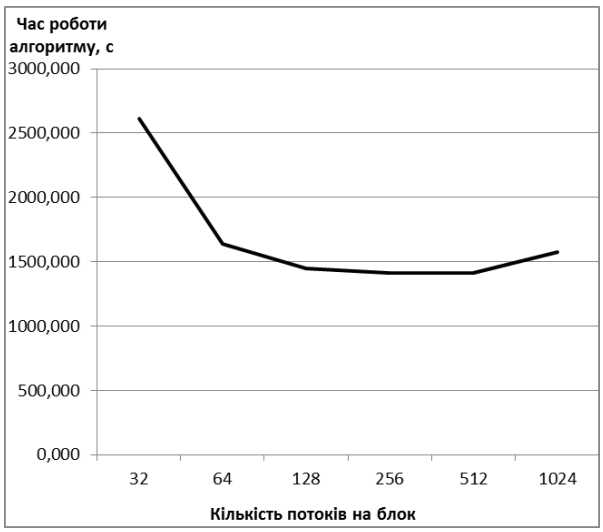

Рис. 4. Залежність часу виконання алгоритму від TPB при N = 2^20

2. Розробка наближеної моделі

2.1. Розробка генератора експерименту. Використання gpusim для дослідження паралельного алгоритму обчислення гравітаційної задачі взаємодії N тіл передбачає розробку експериментального модуля, основною частиною якого є генератор експерименту. Генератор експерименту на основі вхідних даних моделі створює відповідний експеримент [1].

Кожна симуляція експерименту задається конфігураціями Ресурсів Грід (GridResource) та Завдань (Gridlet)[1], відповідно, модель gpusim паралельного алгоритму обчислення гравітаційної задачі взаємодії N тіл має бути представлена за допомогою даних сутностей та їх властивостей.

Виходячи з аналізу експериментальних даних, опису роботи та вихідних кодів алгоритму, можна зробити наступні висновки:

- на час виконання алгоритму, крім вхідних даних, також мають незначний вплив параметри самого алгоритму та апаратно-програмного середовища, у якому він виконується. Точне врахування цих параметрів у моделі не є доцільним, оскільки підвищує її складність, час розробки та зменшує швидкість отримання результатів симуляції;
- оскільки усі обчислення виконуються на GPU, ядра якого мають однакові параметри, то конфігурація ресурсів відповідає гомогенній паралельній системі, яка складається із одного ресурсу із однією ЕОМ із кількістю однакових обчислювальних елементів рівною кількості потоків на блок. Значення рейтингу MIPS обчислювальних елементів повинно бути параметром генератора. Крім того, GridSim для симуляції кожного ресурсу використовує окремий потік, і для підвищення швидкості симуляції замість одного ресурсу із однією ЕОМ та TPB обчислювальних елементів слід використовувати еквівалентну паралельну модель – кількість потоків на блок відповідає кількості однакових ресурсів із однією ЕОМ та одним обчислювальним елементом;

- оскільки кожний потік обчислює вплив одного тіла на інші, то конфігурацію завдань можна представити у вигляді N однакових завдань із розмірами вхідних та вихідних даних, що лінійно залежать від N. Однак усі поточні дані, необхідні алгоритму для роботи, зберігаються у швидкій пам'яті загального користування GPU, тому час, що витрачається на роботу із вхідними та вихідними даними завдання значно менший за час обчислень, тобто модель може знехтувати розмірами вхідних та вихідних даних завдань;
- кількість роботи кожного завдання крім лінійної залежності від N має враховувати операції синхронізації (чим більше потоків використовується у обчисленнях, тим більше часу потребується на їх синхронізацію) та операції переключення контексту (чим більше потоків використовується у обчисленнях, тим менше операцій зміни контексту необхідно для виконання обчислень).

Таким чином, порівняно з попередньою роботою [1], з одного боку, відкинуто частину параметрів, що не впливають на час виконання – а саме параметри CPU-машини , а також витрати на передачу даних в пам'ять GPU. З іншого боку, додано більш складну структуру GPU-ресурсів.

Слід зауважити, що із збільшенням кількості завдань значно зростає час симуляції за рахунок повільної роботи планувальників ресурсів GridSim, а також особливостей Java при створенні великої кількості об'єктів у пам'яті. Для зменшення часу симуляції слід зменшити кількість завдань (Gridlet) та еквівалентно збільшити кількість роботи кожного з завдань.

Виходячи з вищеописаних міркувань, для генератора експерименту пропонується наступна стратегія формування конфігурації завдань:

$$M = \frac{N}{d},$$

$$R = w_1 \cdot \frac{N \cdot \log_2^2(N) \cdot \log_2(TPB)}{TPB},$$

$$S = w_2 \cdot N \cdot TPB,$$

$$L = (N \cdot d + R + S) \cdot K + K_c,$$

де використано позначення:

- M – кількість завдань;
- d – параметр генератора, який призначений для зменшення кількості завдань із еквівалентним збільшенням кількості роботи кожного із завдань;
- R – складова для врахування операцій переключення контексту;
- S – складова для врахування операцій синхронізації;
- w_1 та w_2 – параметри генератора, вагові коефіцієнти R та S відповідно;
- L – кількість роботи кожного завдання;
- K, K_c – параметри генератора, мультиплікативний та адитивний коефіцієнти масштабування величини кількості роботи.

Таким чином, попередньо встановлені такі параметри генератора:

- рейтинг кожного з обчислювальних ядер GPU в одиницях MIPS (million instructions per second): gpuCoreRating;
- параметр зменшення кількості завдань: d = limitationsDivider;
- ваговий коефіцієнт складової для врахування операцій переключення контексту w_1 = smallTPBPenaltyWeight;
- ваговий коефіцієнт складової для врахування операцій синхронізації w_2 = = largeTPBPenaltyWeight;
- значення мультиплікативного та адитивного коефіцієнтів масштабування величини кількості роботи завдань K = = multiplicativeLengthScaleFactor та K_c = = additiveLengthScaleFactor відповідно.

2.2. Розробка планувальника gpusim. Кожний з ресурсів GridSim пропонує користувачу 2 стандартних алгоритми планування завдань у межах ресурсу [12]: First-Come-First-Served(FCFS) [13] та RoundRobin [14]. Якщо сценарій симуляції передбачає використання більше одного ресурсу, то користувач GridSim має самостійно реалізувати високорівневий алгоритм планування. Для підтрим-

ки можливості використання у сценарії симуляції більше одного ресурсу модуль симулятора gpusim був доопрацьований: розроблено планувальник завдань, який реалізує алгоритм RoundRobin. Для зменшення витрат пам'яті сутності завдань створюються з конфігурації по мірі необхідності, а не перед початком симуляції – це суттєво зменшує час симуляції. На рис. 5 показано блок-схему реалізованого алгоритму планування.

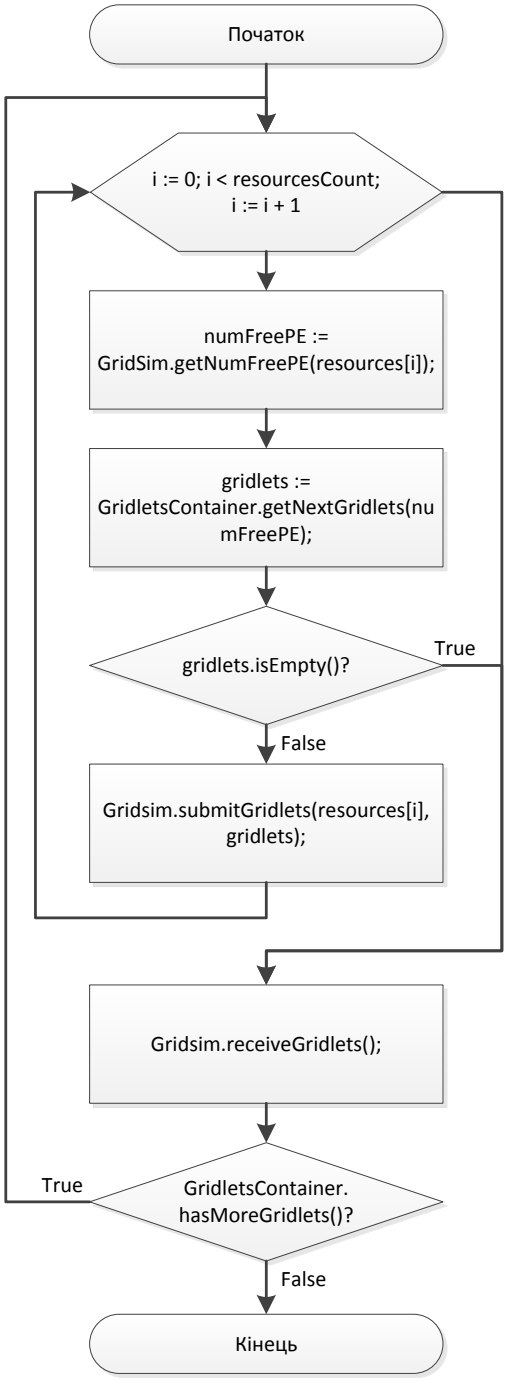

Рис. 5. Блок-схема алгоритму планувальника gpusim

Доки всі завдання, задані конфігурацією завдань, не оброблено, до кожного з ресурсів посилається запит на отримання кількості вільних обчислювальних елементів. Далі контейнер ресурсів створює рівно стільки завдань, скільки у даний час можна розподілити на ресурс, і потім вони відсилаються до ресурсу на виконання, при цьому ресурс у окремому потоці починає виконання завдання одразу після прийому. Після завершення кожного кроку розподілення завдань по ресурсах, виконується їх синхронізація та запис статистичної інформації, далі починається нова ітерація циклу розподілу.

3. Перевірка адекватності та дослідження моделі

3.1. Перевірка адекватності моделі. Для перевірки адекватності моделі використовувались метод та інструментарій, описаний у попередній роботі [1]. Експериментальні дані, отримані раніше при дослідженні алгоритму (N = [2^10..2^20], TPB = [1..1024]), використані як еталонні.

За допомогою модуля оптимізації констант були знайдені точні значення попередньо встановлених параметрів генератора, при яких модель досягає найменшої середньої відносної похибки 43%:

- gpuCoreRating = 1000;
- limitationsDivider = 128;
- smallTPBPenaltyWeigh t= 0.7;
- largeTPBPenaltyWeight = 0.05;
- multiplicativeLengthScaleFactor = 0.1;
- additiveLengthScaleFactor = 0.

На рис. 6 – 9 показано графіки отриманих результатів симуляції (суцільна лінія) у порівнянні із експериментальними даними (пунктирна лінія) для різних значень N та TPB.

Як видно з графіків, найбільша відносна помилка моделі спостерігається при малих значеннях N та TPB. Цю особливість можна пояснити тим, що модель нехтує впливом параметрів алгоритму та програмно-апаратного середовища на час

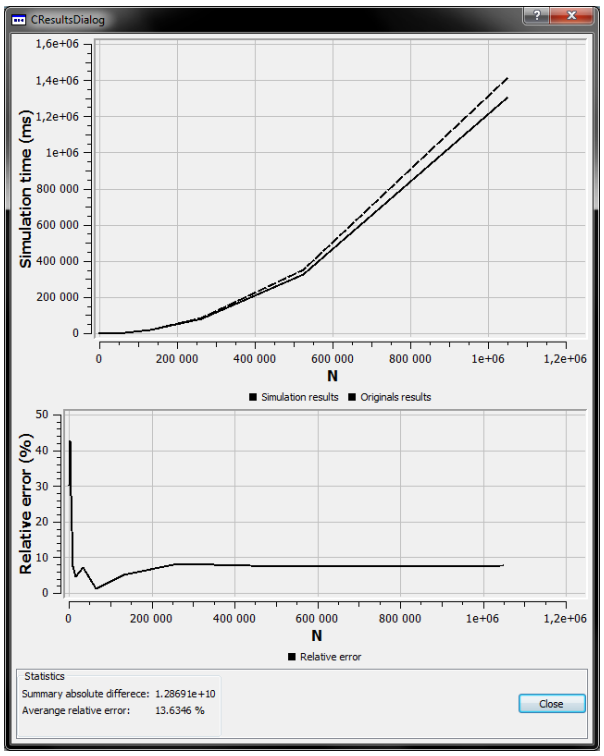

Рис. 6. Порівняння результатів симуляції із експериментальними даними для TPB = 256, N = [2^10, 2^20]

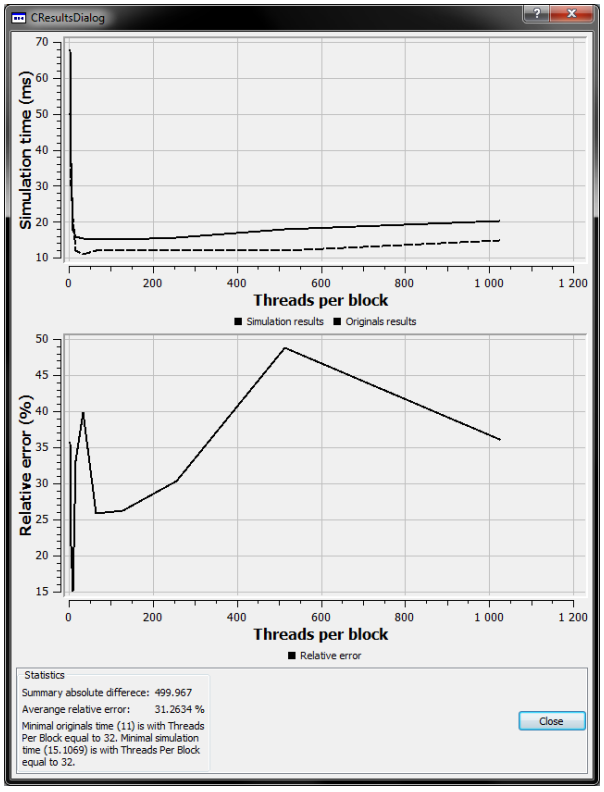

Рис. 7. Порівняння результатів симуляції із експериментальними даними для N = 2^10, TPB = [2, 1024]

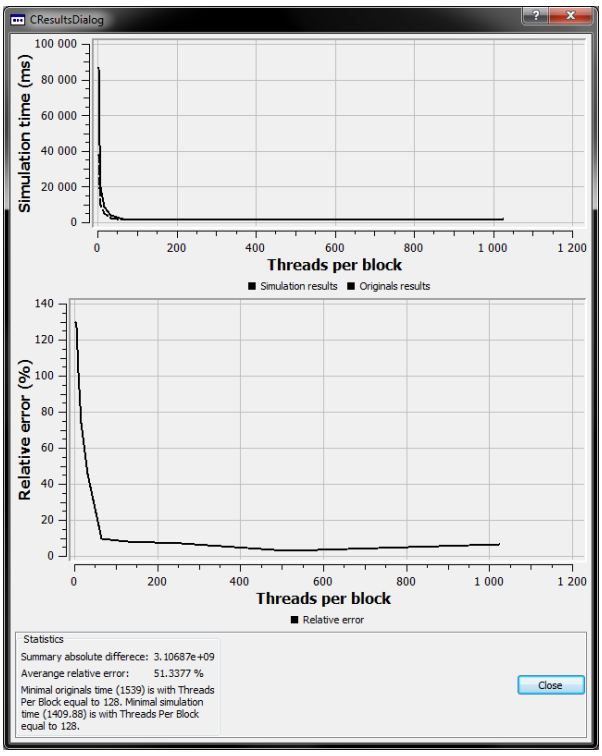

Рис. 8. Порівняння результатів симуляції із експериментальними даними для N = 2^15, TPB = [2, 1024]

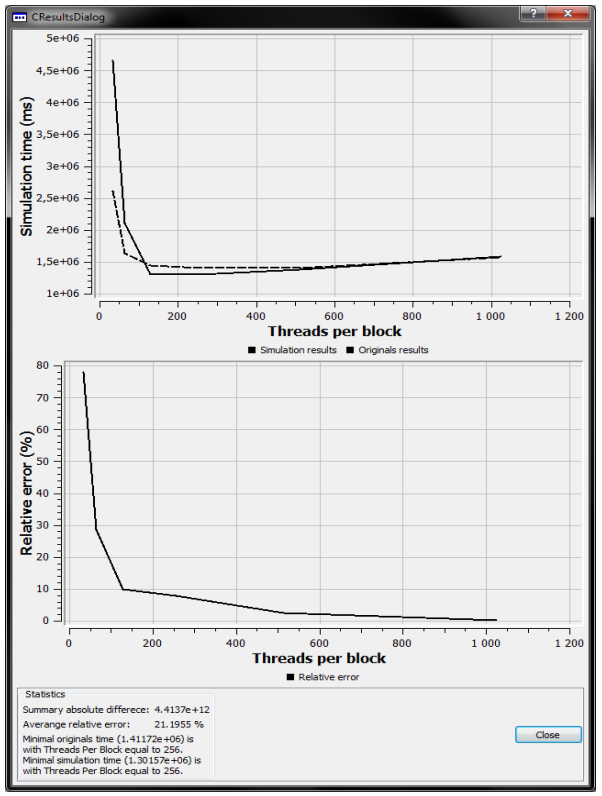

Рис. 9. Порівняння результатів симуляції із експериментальними даними для N = 2^20, TPB = [32, 1024]

роботи алгоритму при малих розмірах вхідних даних. У цій ситуації величина даного впливу порівняна з обсягом обчислень, які виконує алгоритм. Проте, як було відмічено в попередній роботі [1], найбільший інтерес становлять саме великі розміри вхідних даних.

Слід зауважити, що основною задачею моделі є отримання відповіді на питання «Яку кількість потоків на блок слід використовувати в паралельному алгоритмі обчислення гравітаційної задачі взаємодії N тіл?». Зокрема, навіть якщо побудована модель не достатньо точно відтворює час виконання для окремих значень N та ТРВ, але дозволяє правильно відповідати на сформульоване питання, вона вже є корисною для вибору оптимальних значень параметрів.

Розроблена модель зберігає характер залежності часу виконання обчислень від ТРВ при незмінному N: для більших N мінімальне значення часу виконання досягається при більших ТРВ. Іншими словами, точки екстремуму моделі й експериментальних даних збігаються. Крім того, час обрахування моделі значно менший за час проведення експерименту на реальній системі: для N = 2^21 та ТРВ = = [32; 1024] симуляція триває 2,5 хвилини, час експерименту на реальній системі при таких вхідних даних становить 8 годин.

Таким чином, запропонована модель, попри наявність середньої відносної похибки у 43%, за короткий час (3 хвилини для N = 2^21, ТРВ = [32; 1024]) дає коректне значення оптимального параметра ТРВ та може використовуватись у подальших дослідженнях.

3.2. Дослідження моделі. Оскільки для роботи алгоритму із розмірами вхідних даних більших за 2^20 тіл, що обраховуються, потребується багато ресурсів, то наступним кроком є дослідження моделі при великих N для отримання оптимального значення кількості обчислювальних потоків на блок. На рис. 10 – 12 показані графіки залежності часу виконання від ТРВ для N більших за 2^20.

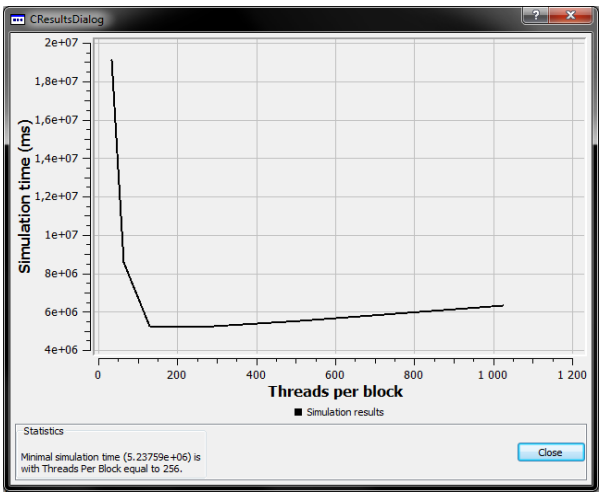

Рис. 10. Залежність часу виконання від ТРВ [32; 1024] для N = 2^21

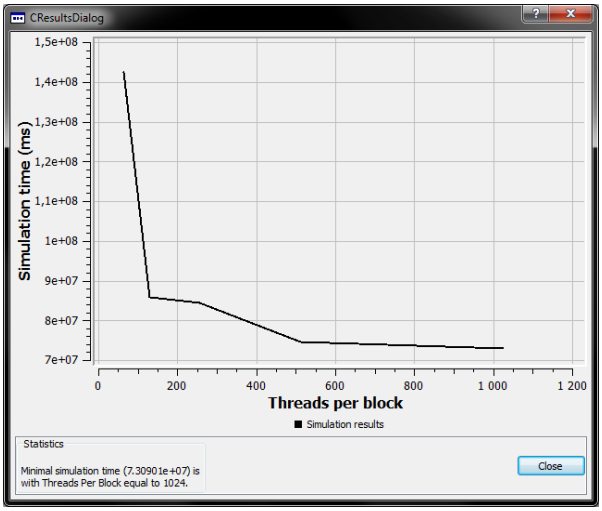

Рис. 11. Залежність часу виконання від ТРВ[64; 1024] для N = 2^23

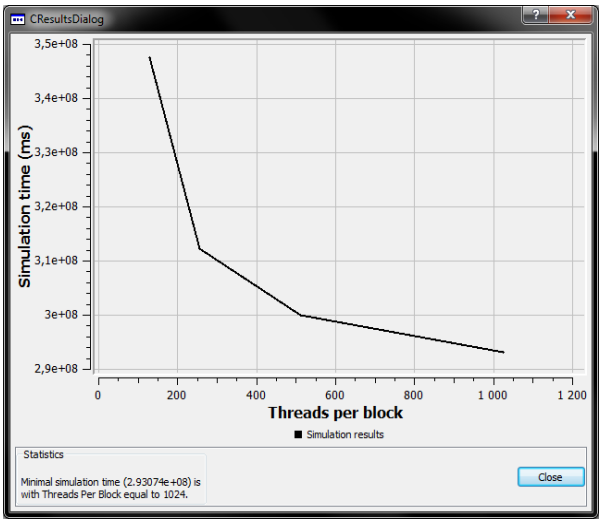

Рис. 12. Залежність часу виконання від ТРВ[128; 1024] для N = 2^24

Аналізуючи отримані графіки можна зробити висновок, що для значень N більших за 2^20 оптимальними будуть наступні значення TPB:

- якщо 2^20 <= N < 2^22, оптимальне TPB = 256;
- якщо 2^22 < N < 2^23, оптимальне TPB = 512;
- якщо N > 2^23, оптимальне TPB = = 1024.

Слід відмітити, що використання алгоритму для великих значень N потребуватиме внесення певних змін, щоб не перевантажувати пам'ять GPU.

Висновки

У роботі проведено дослідження та моделювання виконання паралельного алгоритму на відеографічному мультипроцесорі на прикладі гравітаційної задачі взаємодії N тіл. Виявлено, що в залежності від розміру вхідних даних для досягнення мінімального часу виконання слід обирати різну величину кількості потоків CUDA на блок. Для подальшого дослідження цієї особливості було запропоновано, розроблено та налаштовано наближену модель алгоритму і паралельної системи, на якій він виконується для середовища моделювання гетерогенних паралельних систем gpusim.

Розроблена модель, проте наявність середньої відносної похибки у 43%, за короткий час дає коректну відповідь на питання «Яку кількість потоків на блок слід використовувати в паралельному алгоритмі обчислення гравітаційної задачі взаємодії N тіл?». Моделювання також дозволяє суттєво прискорити процес отримання оптимальних значень параметрів: наприклад, для N = 2^21 оптимальне значення TPB = 256 отримане за 3 хвилини, порівняно з 8 годинами, необхідними для експериментального підтвердження цього значення.

Експерименти із моделлю для великих значень N (більших за 2^20) дали наступні результати:

- якщо 2^20 <= N < 2^22, оптимальне TPB = 256;
- якщо 2^22 < N < 2^23, оптимальне TPB = 512;
- якщо N > 2^23, оптимальне TPB = = 1024.

Подальші дослідження в даному напрямку передбачають підвищення точності моделі, оптимізацію симулятора для прискорення моделювання при великій кількості завдань та дослідження можливості використання інтелектуальних алгоритмів для автоматизації розробки наближених моделей для gpusim.

1. *Оконський І.В., Дорошенко А.Ю., Жереб К.А.* Інструментальні засоби моделювання гетерогенних середовищ заснованих на відеографічних прискорювачах // Проблеми програмування. – 2013. – № 1. – С. 107–115.
2. *Sulistio A., Cibej U., Venugopal S., et al.* A toolkit for modelling and simulating data Grids: an extension to GridSim // Concurrency and Computation: Practice & Experience. – 2008. – Vol. 20, N 13. – P. 1591–1609.
3. *Sverre J. Aarseth.* Gravitational N-body simulations. – Cambridge University Press, 2003. – 413 p.
4. *Hamada T., Nitadori K.* 190 TFlops Astrophysical N-body Simulation on a Cluster of GPUs // Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society, Washington, DC, USA. – 2010. – P. 1–9.
5. *Bédorf J., Gaburov E., Zwart S.P.:* A sparse octree gravitational N-body code that runs entirely on the GPU processor // Journal of Computational Physics. – 2012. – Vol. 231, N 7. – P. 2825–2839.
6. *Belleman R.G., Bédorf J., Portegies Zwart S.F.* High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA // New Astronomy– 2008. – Vol. 13, N 2. – P. 103–112.
7. *Hamada T., et al.* 42 Tflops hierarchical n-body simulations on GPUs with applications in both astrophysics and turbulence // Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. – 2009. – P. 62.
8. *Anderson J.A., Lorenz C.D., Travesset A.* General purpose molecular dynamics simulations fully implemented on graphics

processing units // Journal of Computational Physics. – 2008. – Vol. 227, N 10. – P. 5342–5359.
9. *Makino J., Aarseth S.J.* On a Hermite integrator with Ahmad-Cohen // Publications of the Astronomical Society of Japan – 1992. – Vol. 44, N 2. – P. 141–151.
10. *GeForce* GTX 650 Specifications [Электроний ресурс]. – Режим доступу: http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-650/specifications. – 01.11.2012 р.
11. *CUDA API REFERENCE MANUAL* [Электроний ресурс]. – Режим доступу: http://docs.nvidia.com/cuda/pdf/CUDA_Toolkit_Reference_Manual.pdf – 01.11.2012 р.
12. *ResourceCharacteristics* (GridSim 5.0 beta API Specification) [Электроний ресурс]. – Режим доступу: http://www.buyya.com/gridsim/doc/api/gridsim/ResourceCharacteristics.html – 01.11.2012 р.
13. *First-come, first-served* – Wikipedia, the free encyclopedia [Электроний ресурс]. – Режим доступу: http://en.wikipedia. org/wiki/First-come,_first-served. – 01.11.2012 р.
14. *Round-robin* (алгоритм) – Википедия [Электроний ресурс]. – Режим доступу: http://ru.wikipedia.org/wiki/Round-robin_(алгоритм). – 01.11.2012 р.

Одержано 16.10.2012

Про авторів:

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень Інституту
програмних систем НАН України,

Оконський Ілля В'ячеславович,
студент факультету інформатики
та обчислювальної техніки,
кафедри автоматики і управління
в технічних системах НТУУ “КПІ”,

Жереб Костянтин Анатолійович,
кандидат фізико-математичних наук,
науковий співробітник,

Бекетов Олексій Геннадійович,
аспірант.

Місце роботи авторів:

Національний технічний університет
України "КПІ"
03056, Київ-56,
Проспект Перемоги, 37
Тел.: (044) 236 7989,

Інститут програмних систем
НАН України,
03680, Київ-187,
Проспект Академіка Глушкова, 40.
Тел.: (044) 526 1538.

E-mail: dor@isofts.kiev.ua,
logrus.work@gmail.com
zhereb@gmail.com
beketov.oleksii@gmail.com