ЦИФРОВА ЕКОНОМІКА ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ
Digital Economy and Information Technologies

УДК 004.4

Данило Ушенін,
Сергій Сабанов,
Анатолій Переверзєв,

кафедра інформаційних технологій
Запорізький інститут економіки та інформаційних технологій
Запоріжжя, Україна

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПАРАЛЕЛЬНИХ ОБЧИСЛСЕНЬ НА GPU В СЕРЕДОВИЩІ ADOBE AFTER EFFECTS

Анотація – У роботі розглядаються особливості використання паралельних обчислень на базі графічної підсистеми комп'ютера у середовищі Adobe After Effects. Аналіз показав, що Adobe After Effects підтримує декілька способів використання GPGPU (General-Purpose computing on Graphics Processing Units), зокрема це CUDA, OpenCL та Metal. Достовірну оцінку доцільності використання GPGPU в даному випадку забезпечить урахування двох складових: відомої проблеми розпаралелювання програм, пов'язаної з забезпеченням правильної послідовності взаємодій між різними обчислювальними процесами та координації ресурсів, що розділяються між ними, а, також, дихотомії програм, що використовують CUDA, OpenCL abo Metal, представленої стандартним кодом для CPU та кодом для GPU (kernel).
В даній статті наведено дослідження факторів, які впливають на ефективність використання паралельних обчислень на GPU та вплив цих факторів на результати роботи різних технологій GPGPU при виконанні типових операцій у середовищі Adobe After Effects. Результати дослідження можуть бути використані на етапі конфігурування графічної підсистеми комп'ютера для роботи з середовищем.

Ключові слова – *Adobe After Effects, CUDA, GPGPU, GPU, Metal, OPENCL, паралельні обчислення***.**

I. ВСТУП

З розвитком Інтернету і зростанням популярності різноманітних відеоплатформ, все більше користувачів починають так чи інакше мати справу з обробкою відео. Для цього існує багато спеціалізованих програм, одною з яких є Adobe After Effects – програма для редагування відео, створення анімацій і різних ефектів.

Зазвичай Adobe After Effects для обробки відео або зображень використовує CPU. При застосуванні більшості ефектів для кожного пікселя послідовно виконується однаковий алгоритм для знаходження кінцевого результату. В залежності від роздільної здатності відео, навіть в одному кадрі загальна кількість пікселів може досягати декількох мільйонів, тому обробка відео може займати багато часу, особливо при необхідності виконання складних алгоритмів. Для прискорення обробки відео, в Adobe After Effects є можливість використання паралельних обчислень на GPU, що при певних обставинах може зменшити час обробки відео у декілька разів, але досягнути такої ефективності можна далеко не на всіх задачах. Така «неуніверсальність» графічного процесора обумовлена специфікою його побудови. Одна з головних відмінностей від CPU – об'єднання обчислювальних ядр в блоки, що мають загальні елементи. В процесі розрахунку не можна виділити тільки одне ядро, буде виділено цілий блок. Крім того через відносно простий склад логічних блоків та загальних регістрів, GPU досить посередньо оброблює розгалуження, та складну логіку в алгоритмах.

Сучасні відеокарти оснащені універсальними графічними процесорами, що підтримують набори інструкцій усіх типів шейдерів, тобто можна стверджувати, що шейдерна архітектура стала уніфікованою. Очевидні позитивні риси уніфікації дещо нівелюються існуванням різних наборів інструкцій для GPU різних виробників.

В залежності від операційної системи, виробника та моделі відеокарти існує декілька способів використання GPU в середовищі Adobe After Effects. До них відносяться: CUDA, OpenGL, OpenCL та Metal [1]. В роботі проведено оцінку ефективності використання цих технологій на однотипних задачах та визначено фактори, які суттєво впливають на продуктивність паралельних обчислень засобами графічної підсистеми. Визначення цих факторів може допомогти користувачам Adobe After Effects як на етапі конфігурування графічної підсистеми під час вибору апаратних компонентів, так і безпосередньо при роботі в середовищі під час виконання його налаштувань.

II. МОДЕЛІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

Уніфікація шейдерних архітектур дозволяє виділити в технічній документації для різних технологій GPGPU спільні основні моменти процесу обміну даними між робочим елементом (потоком) та CPU [2, 3, 4]. Видно, що такий процес проходить з використанням глобальної пам'яті, яку сумісно використовують і GPU, і центральний процесор.

Програма, яка використовує CUDA, OpenCL або Metal складається з двох частин – звичайного коду для CPU та kernel-коду, який буде виконуватися на GPU. Всі три технології GPGPU мають схожий принцип роботи, який можна змоделювати у вигляді таблиці (одновимірної, двовимірної або тривимірної), з наступним розбиттям на декілька блоків. Блок має багато робочих елементів (потоків), які будуть виконуватися паралельно і оброблювати один елемент таблиці виконуючи kernel-код на GPU.

Слід зауважити, що термінологія, щодо частин задачі та елементів в різних технологіях дещо різниться:

Таблиця 1.Порівняння термінів різних технологій GPGPU

Назва частини	CUDA	OpenCL	Metal
Вся задача	Grid	Grid	Grid
Блоки, на які розділяють всю задачу	Block	Work group	Threadgroup
Найменший елемент задачі, який буде обробляти kernel-код в окремому потоці	Thread	Work item	Thread

Блоки, на які ділиться задача мають бути одного розміру, при неможливості розбити задачу таким чином, кількість блоків округляють в більшу сторону і, як наслідок, з’являються зайві елементи. В [3] наведено спосіб подолання проблеми: в цьому випадку в kernel потрібно додати захисний код, щоб гарантувати, що він не буде виконуватися за межами даних.

Координати елементу (потоку) у глобальній сітці можна знайти за допомогою вбудованих змінних та функцій kernel, що трохи відрізняються у різних технологіях (таблиця 2).

Таблиця 2. Вбудовані змінні для орієнтування в глобальній сітці для різних технологій GPGPU

CUDA-kernel	
Змінна	Опис
gridDim	Розміри сітки
blockIdx	Індекс блоку всередині сітки
blockDim	Розміри блоку
threadIdx	Індекс потоку в блоці
OpenCL-kernel	
Функція	Опис
get_work_dim	Розмірність
get_local_size	Кількість локальних робочих елементів
get_local_id	Локальний ідентифікатор робочого елемента
get_num_groups	Кількість робочих груп
get_group_id	Ідентифікатор робочої групи
Metal-kernel	
Змінна	Опис
threadgroup_position_in_grid	Положення групи потоків у сітці
thread_position_in_threadgroup	Положення потоку в групі потоків
threads_per_threadgroup	Кількість потоків на групу потоків

Для опису механізму доступу до потоків побудуємо модель, що описує положення одного потоку (рисунок 1). Координати блоку на глобальній сітці в нашому випадку (1,1); координати потоку в блоці – (2,1).

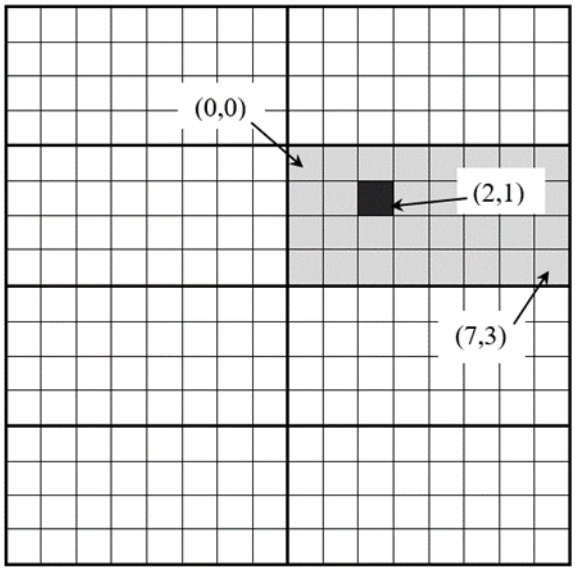

Рис. 1. Координати потоку в глобальній сітці

Тоді координати потоку в глобальній сітці можна визначити, помноживши розмір блоку на позицію цього блоку в глобальній сітці а потім додавши позицію потоку в цьому блоці:

$$\begin{cases} x = block\_width * block\_in\_grid.x + thread\_in\_block.x \\ y = block\_height * block\_in\_grid.y + thread\_in\_block.y \end{cases} \quad (1)$$

Робота в кожній робочій групі виконується незалежно від інших, порядок виконання невизначений, і не існує засобів синхронізації між різними робочими групами, але в межах однієї робочої групи є засоби синхронізації робочих елементів. Також треба брати до уваги те, що відеокарта має обмеження на максимальний розмір робочої групи.

Архітектура GPU побудована на основі масштабованого масиву багатопоточних мультипроцесорів (Streaming Multiprocessors). Коли програма виконується на GPU, викликається сітка kernel, блоки сітки перераховуються та розподіляються між мультипроцесорами. Потоки одного блоку потоків виконуються одночасно на одному мультипроцесорі, також можуть одночасно виконуватися декілька блоків потоків. Коли блоки потоків завершують роботу, на звільнених мультипроцесорах запускаються нові блоки.

Для використання kernel потрібно лише кілька спеціалізованих функцій, переважно для роботи з даними і для запуску самого kernel. В разі використання, наприклад, CUDA це може виглядати так (рис. 2):

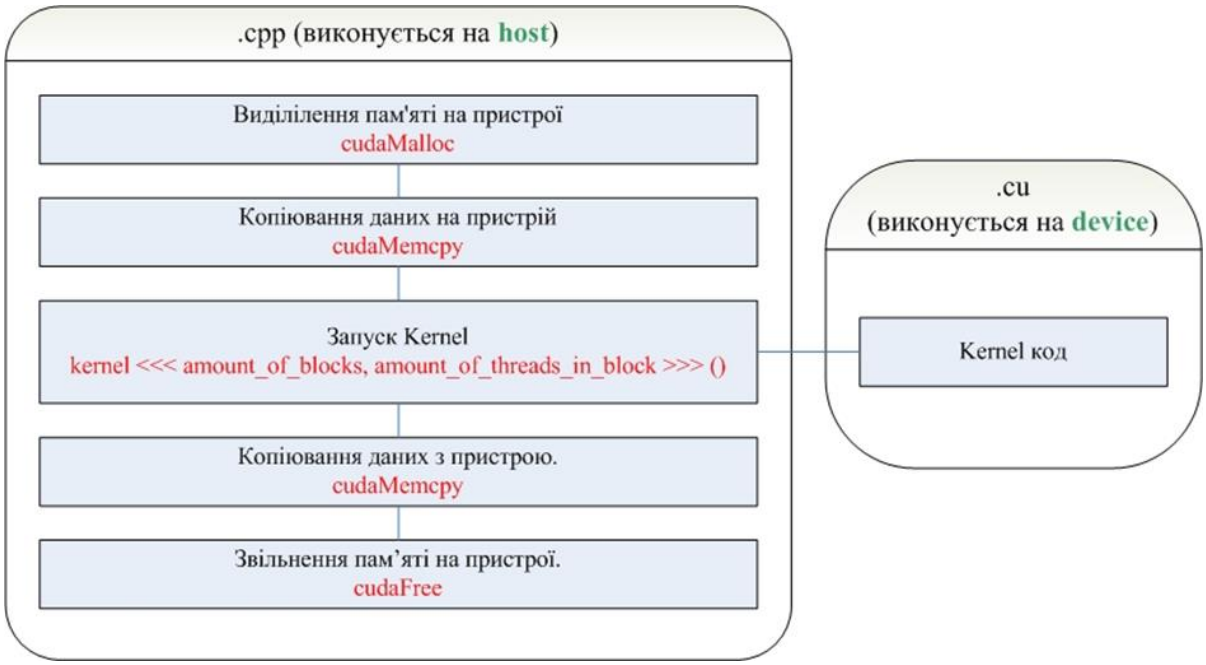

Рис. 2. Запуск CUDA-додатку

В порівнянні з CUDA-додатками, для використання OpenCL спочатку треба зробити досить велику кількість налаштувань.

III. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

При виборі технології GPGPU для Adobe After Effects варто відзначити що їх доступність обмежена операційною системою, виробником та моделлю відеокарти, але при деяких обставинах користувачеві може бути доступно одразу декілька способів

використання GPGPU.

Adobe After Effects реалізований для роботи в середовищі двох операційних систем: Windows та MacOS. Остання традиційно вважається лідером в плані сумісності з ПЗ, спеціалізованого на обробці мультимедіа, і може надати користувачу повну підтримку технологій GPGPU (таблиця 3).

Таблиця 3. Підтримка технологій GPGPU в середовищі різних ОС

	Windows	MacOs
CUDA	+	Підтримуються тільки старі версії (до CUDA 10.2)
OpenCL	+	+
Metal	-	+

Наступним фактором, що його прийдеться враховувати під час конфігурування системи пов’язаний з апаратною реалізацією паралельних обчислень різними вендорами відеокарт. Щоправда вибір тут обмежений двома виробниками GPU – Nvidia та Amd (таблиця 4).

Таблиця 4.Реалізація технологій GPGPU виробниками графічних процесорів

	Nvidia	Amd
CUDA	+	-
OpenCL	+	+
Metal	-	+

В ході дослідження було використано систему, що працює під управлінням ОС Windows з графічною підсистемою від Nvidia – ASUS Vivobook Pro. Конфігурація:

- ✓ CPU – Intel Core i5-12500H (18 MB Intel Smart Cache);
- ✓ відео – NVIDIA GeForce RTX 3050 (4GB);
- ✓ RAM – 16 ГБ LPDDR5.

Тип (а значить і швидкодія накопичувачів) у випадку тестування продуктивності GPGU не має ніякого значення, більш важливими є такі параметри нашої системи:

- ✓ швидкість обміну по шині PCI-e x16 (для Intel Core i5-12500H це Gen 4 [6]) – 31.5 GB/s
- ✓ швидкість обміну по шині GPU – GPU RAM (для розрядності 128 біт) – 224 GB/s
- ✓ продуктивність ОЗП (обмежується процесором до 5200 MT/s) [6] – 41,6 GB/s
- ✓ кількість ядер CUDA в складі GPU – 2560 [5].

Хоча швидкість обміну графічного процесора з пам’яттю відеокарти є найвищою в системі, ця перевага нівелюється каналом між ОЗП та GRAM, що представлений шиною PCI-e з швидкістю, що менша в сім разів. Для вирішення типової задачі масового паралелізму з великою кількістю чисел засобами GPU, їх необхідно спочатку передати в пам’ять графічної карти по шині PCI-e. Система, що використовує процесор Intel Core i5-12500H, має четверту генерацію шини [6], швидкість якої при використанні стандартних 16-ти ліній складає 31.5 GB/s. Відеокарта виконає обробку даних дуже швидко, але після обробки дані потрібно буде знов передавати в RAM, тобто може

виявитися, що набагато ефективніше було б просто передати дані більш продуктивним каналом в CPU, і, незважаючи на значний програш центрального процесора у швидкодії, він виконає обробку швидше, ніж у випадку, якби вони передавалися в GPU.

Враховуючи обмеження, що наведені в таблицях 3 та 4 та конфігурацію тестової системи, було виконано порівняння продуктивності виконання простих обчислень для двох доступних технологій: CUDA та OpenCL.

Беручи до уваги модель обчислень, що її використовує GPU, було підібрано умови, за яких кількість вхідних даних перевищувала б розміри глобальної сітки потоків з їхньої обробки та гарантовано б переповнювала кеші всіх рівнів. В якості задачі, що відповідає таким умовам може служити обробка масиву з десяти мільйонів чисел. Враховуючи спеціалізацію GPU на простих операціях, було обрано найпростіший алгоритм збільшення кожного числа на одиницю.

Класичний обхід елементів у циклі для GPU буде відрізнятися використанням змінної threadIdx, яку без оголошення надає система. Індекс (або номер потоку) відеокарта передає, як статичну змінну, що може характеризуватися кількома вимірами – x, y, z. Для запуску відразу великої кількості паралельних потоків їх доведеться розбити на блоки, максимальний розмір яких залежить від відеокарти, а індекс елемента, для якого ведеться обчислення, отримуємо, як вказано раніше в описі моделі (i = block_width * block_in_grid.x + thread_in_block.x).

Результатом буде багато паралельно працюючих потоків, що виконують однаковий код, але з різними індексами і різними даними (SIMD).

Для кількісного порівняння продуктивності OpenCL та CUDA, крім загального часу виконання задачі, були виміряні й додаткові параметри для кращого розуміння того, на що саме витрачається переважна більшість часу при використанні GPGPU. Результати порівняння наведені в таблиці 5.

Таблиця 5.Порівняння продуктивності OpenCL та CUDA

	OpenCL	CUDA
Виділення пам’яті на GPU, ms	0.03	1.01
Копіювання даних на GPU, ms	16.81	16.15
Виконання обчислень на GPU, ms	0.09	0.10
Копіювання даних з GPU, ms	12.53	13.79
Звільнення пам’яті на GPU, ms	0.27	0.37
Різні налаштування (більшість з таких налаштувань виконується лише один раз), ms	268.22	0
Загальний час виконання, ms	297.95	31.42

Результати дуже різнилися при повторних вимірах, але загальне співвідношення часу на різних етапах задачі залишалося більш-менш однаковим. Основні етапи задачі (копіювання даних на GPU, виконання обчислень на GPU, копіювання даних з GPU займали приблизно однакову кількість часу для CUDA та OpenCL, що цілком логічно, бо визначається більше апаратною складовою, аніж програмною.

Щодо принципіальної різниці в плані затрат часу на налаштування, то, як це не дивно, ними можна знехтувати, бо більшість налаштувань для використання GPGPU

необхідно виконати лише один раз; їх можна винести у спеціальну функцію GPUDeviceSetup, яка запускається при першому застосуванні плагінів Adobe After Effects.

З цього можна зробити висновок що для даної задачі основну кількість часу займали саме обмін даними між CPU та GPU (якщо не брати до уваги ті самі "налаштування"). Хоча на відеокарті можна одночасно робити безліч обчислень, через низьку швидкість обміну даними між CPU та GPU це буде неефективно якщо з цими даними треба зробити всього декілька операцій.

По відношенню до задач, що вирішуються в Adobe After Effects отримані результати означають, що використання GPGPU підходить для випадків, де обчислення для одного пікселя не залежать від обчислень для іншого пікселя. Більшість ефектів в Adobe After Effects можна розподілити за їх призначенням: ефекти для корекції кольору, деформації, генерації різноманітних об'єктів, роботи з 3D, симуляції різних явищ, створення переходів, роботи з текстом, тощо. Незважаючи на те що всі ефекти використовують різні алгоритми обчислень, різновиди графічної обробки значно різняться по відношенню до використання обчислень на GPU. Зокрема, ефекти для роботи з кольором зазвичай підходять для цього, адже в таких задачах не потрібна синхронізація обчислень, в протилежність цьому ефекти деформації, – навпаки не дозволяють ефективно використовувати GPGPU через велику кількість операцій обміну між сутностями «host» та «device».

IV. ВИСНОВКИ

Adobe After Effects підтримує декілька способів використання GPGPU: CUDA, OpenCL, Metal. Всі три технології мають схожий принцип роботи, архітектура програмного забезпечення, що створюється з застосуванням даних технологій представлена двома частинами – звичайного коду для CPU (host-частина) та kernel коду, який буде виконуватися на GPU (device).

Для використання можливостей CUDA та OpenCL задачу потрібно представити у вигляді таблиці (розмірністю від одного до трьох), яку потім розбивають на блоки, що містять робочі елементи (потоки), що будуть виконуватися паралельно і оброблювати один елемент таблиці виконуючи kernel код на GPU.

Обмеженість синхронізації обчислень межами одного блоку та відносно низька швидкість обміну даними між CPU та GPU можуть суттєво впливати на ефективність використання GPGPU в ситуаціях, коли з даними потрібно зробити незначну кількість операцій, або якщо даних для обробки не дуже багато.

Використання GPGPU в середовищі Adobe After Effects підвищує ефективність роботи лише для певного кола задач, якими зокрема є ефекти для роботи з кольором, що можна пояснити відсутністю необхідності синхронізації обчислень CPU та GPU. З іншого боку, ефекти деформації, навпаки не дадуть змогу раціонально задіяти ресурси GPU через велику кількість операцій обміну між графічним та центральним процесорами, коли оверхед з’являється на пересиланні даних з ОЗП в пам'ять відеокарти.

Таким чином під час вибору обладнання для роботи з Adobe After Effects урахування підтримки CUDA, OpenCL чи Metal явно недостатньо, бо досить велика частина обробки ефектів буде визначатися такими параметрами, як пропускна спроможність пам'яті, версія реалізації шини PCI Express та імплементація того, як передаватимуться дані на карту (паралельними потоками чи ні).

V. ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Отримані результати дозволяють спробувати переглянути традиційний підхід до розробки спеціалізованих плагінів Adobe After Effects, що за замовчанням має на увазі використання технологій GPGU без уваги на доцільність ускладнення коду. В цьому сенсі перспективним виглядає дослідження ефективності використання ресурсів GPU вже не на синтетичному тесті з абстрактним великим масивом, а в реалізації коду на прикладі плагінів, що забезпечують різні ефекти при роботі з середовищем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] GPU and GPU driver requirements for After Effects [Електронний ресурс] / Adobe. – Режим доступу: www. URL: https://helpx.adobe.com/ua/after-effects/using/basics-gpu-after-effects.html

[2] CUDA Toolkit Documentation [Електронний ресурс] / – Режим доступу: www. URL: https://docs.nvidia.com/cuda/index.html

[3] Calculating Threadgroup and Grid Sizes [Електронний ресурс] / Apple. Documentation. Metal. Compute Passes. – Режим доступу: www. URL: https://developer.apple.com/documentation/metal/calculating_threadgroup_and_grid_sizes

[4] Lee Howes. OpenCL™ Parallel computing for CPUs and GPUs [Електронний ресурс] / Advanced Micro Devices. – Режим доступу: www. URL: https://developer.amd.com/OpenCL_Parallel_Computing_for_CPUs_and_GPUs_201003.pdf

[5] Архитектура NVIDIA. GeForce RTX 3050 [Електронний ресурс] / NVIDIA. Ampere. – Режим доступу: www. URL: https://www.nvidia.com/ru-ru/geforce/graphics-cards/30-series/rtx-3050/

[6] Intel® Core™ i5-12500H Processor [Електронний ресурс] / Intel. – Режим доступу: www. URL: https://ark.intel.com/content/www/us/en/ark/products/96141/intel-core-i512500h-processor-18m-cache-up-to-4-50-ghz.html

[7] MrUSmith. Nvidia Ampere – самое главное о новой архитектуре видеокарт [Електронний ресурс] / DNS клуб. Блог. Видеокарты – Режим доступу: www. URL: https://club.dns-shop.ru/blog/t-99-videokartyi/41823-nvidia-ampere-samoe-glavnoe-o-novoi-arhitekture-videokart/

[8] G. Osipov. Вычисляем на видеокартах. Технология OpenCL. Часть 0. Краткая история GPGPU [Електронний ресурс] / Хабр – Режим доступу: www. URL: https://habr.com/ru/company/yandex_praktikum/blog/575484/

[9] Д. Павлов. Сравнение времени выполнения алгоритма на CPU и GPU [Електронний ресурс] / Хабр – Режим доступу: www. URL: https://habr.com/ru/post/525892/

Отримано 12.10.2022 р.

FEATURES OF USING PARALLEL CALCULATIONS ON GPU IN ADOBE AFTER EFFECTS ENVIRONMENT

Danilo Ushenin,
Sergey Sabanov,
Anatoly Pereverzev,

Department of Information Technologies
Zaporizhzhya Institute of Economics and Information Technologies
Zaporizhzhia, Ukraine

Annotation – The authors look at the features of different parallel calculations based on the graphics subsystem of the computer in the middle of Adobe After Effects. The analysis showed that Adobe After Effects supports a number of ways to use GPGPU (General-Purpose computing on Graphics Processing Units), especially CUDA, OpenCL and Metal. However, it is necessary to investigate the problems of parallelization of programs that are related to the security of the correct sequence in interdependence between different processes and the coordination of resources that are divided between them, as well as the dichotomy of programs that use CUDA, OpenCL or Metal.

This article examines the factors that affect the efficiency of parallel computing on the GPU and the impact of these factors on the performance of various GPGPU technologies when performing typical operations in the Adobe After Effects environment. The results of the research can be used at the stage of configuring the graphic subsystem of the resources that are divided between them, as well as the dichotomy of programs that use CUDA, OpenCL or Metal. This article examines the factors that affect the efficiency of parallel computing on the GPU and the impact of these factors computer to work with the environment.

Keywords *- Adobe After Effects, CUDA, GPGPU, GPU, Metal, OPENCL, parallel calculations.*

Received 12.10.2022