Збірник наукових праць Центру воєнно-стратегічних досліджень Національного університету оборони України. 2024. № 2(81), с. 74–85.

АЛГОРИТМИ ТА МЕТОДИ СИНГУЛЯРНОГО РОЗКЛАДУ МАТРИЦЬ НА ГРАФІЧНИХ ПРОЦЕСОРАХ З CUDA / ALGORITHMS AND METHODS FOR SINGULAR VALUE DECOMPOSITION OF MATRICES ON CUDA-ENABLED GPUS

Сухарський С. С. / Sukharskyi S. S.
Національний університет "Києво-Могилянська академія" / National University of "Kyiv-Mohyla Academy"
04655, Київ, вул. Григорія Сковороди, 2, каф. Мережних технологій, (044) 425-77-23
E-mail: s.sukharskyy@ukma.edu.ua

Singular Value Decomposition (SVD) is a fundamental tool in linear algebra with a wide range of applications, including data processing and machine learning. With the advancement of Graphics Processing Units (GPUs), optimizing computational methods for SVD has become a pressing task. This study introduces new approaches to bidiagonalization and diagonalization of matrices using Householder transformations and Givens rotations, effectively leveraging GPU capabilities. Developed algorithms demonstrate high speed and accuracy while minimizing memory usage. Experiments revealed that proposed bidiagonalization algorithm works quite well, while optimizing diagonalization by reducing block synchronization delays is the main challenge. Future work will focus on improving these aspects to enhance performance on large matrices.

Сингулярний розклад (SVD), який описується формулою $A = U\Sigma V^T$, де A вхідна матриця $m \times n$, U ортогональна матриця розміру $m \times m$, Σ діагональна матриця $m \times n$ з невід'ємними значеннями розташованими за спаданням, а V ортогональна матриця $n \times n$, є фундаментальним інструментом у лінійній алгебрі та чисельних методах, з широким спектром застосувань: від обробки сигналів і зображень до машинного навчання та аналізу великих даних. Зі зростанням обсягів даних і складності моделей виникає необхідність у більш ефективних обчислювальних методах. Графічні процесори (GPU) з технологією CUDA надають можливість значно прискорити обчислення завдяки їх масовому паралелізму. Тому розробка ефективних алгоритмів SVD для GPU є вкрай актуальною для підвищення продуктивності в різних наукових та інженерних областях. Багато дослідників розробляли паралельні версії традиційних алгоритмів SVD, таких як метод Джейкобі або QR-алгоритм, для виконання на GPU [3]. Наприклад Лагабар та Нараянан в своєму дослідженні [3] використовують метод Хаусхолдера та неявно зміщений QR алгоритм. При чому, серію трансформацій Хаусхолдера виконують лише на графічному процесорі, а діагоналізацію з залученням центрального процесора для контролю ходу виконання. В своєму дослідженні я використовую інший підхід, де саме бідіагоналізація виконується з контролем ходу виконання на центральному процесорі, а друга частина вже відбувається повністю на відеокарті, щоб максимально ефективно використати можливості відповідних процесорів. Частина дослідників сфокусувалась на рандомізованих алгоритмах, що обчислюють наближений SVD для великих матриць [5]. Цей підхід дозволяє пришвидшити обчислення, проте підходить не для всіх задач. Також є дослідження блокових алгоритмів сингулярного розкладу, що обробляють матриці частинами [4]. Цей підхід є перспективним, проте не розкриває проблему синхронізації даних між блоками, над чим я працюю в дослідженні.

Зміст цього дослідження полягає в розробці та реалізації оптимізованого алгоритму SVD для графічних процесорів, який перевершує існуючі рішення за критеріями часу виконання, точності обчислень та ефективності використання пам'яті. Рішення включатиме нові підходи до паралелізації обчислень, оптимізацію використання архітектури GPU, а також вдосконалення чисельних методів для зменшення похибок і

уникнення втрат точності. Результати дослідження можуть зробити значний внесок у галузь високопродуктивних обчислень і відкрити нові можливості для застосування SVD у реальних задачах, що вимагають обробки великих обсягів даних у режимі реального часу.

Станом на зараз розроблено та реалізовано метод зведення матриці до двоідагонального вигляду, що є першою частиною сингулярного розкладу. Метод використовує серію трансформацій Хаусхолдера, та максимізує використання графічного процесора для проведення необхідних обчислень. В алгоритмі матриця завантажується на відеокарту і усі наступні операції виконуються з даними лише на відеокарті. При цьому, контроль потоку виконання операцій відбувається на центральному процесорі. По ходу алгоритму ми проходимо усі стовпчики і рядки матриці занулюючи потрібні елементи. Для цього ми будуємо вектори Хаусхолдера і проводимо необхідні операції з ними на графічному процесорі. За цією темою була опублікована наукова стаття [1], де можна детальніше ознайомитись з самим алгоритмом.

Також, розроблено та реалізовано метод діагоналізації матриці, з використанням ітеративного процесу, який будує матриці повороту Ґівенса та застосовує їх до проміжної матриці для занулення значень, а також до додаткових матриць-множників зліва та справа. Процес ітераційний, оскільки після множення матриці повороту на основну у нас занулюються одні значення над або під діагоналлю, залежно від того, з якими ми працюємо під час відповідного проходження матриці, та "засмічуються" протилежні значення, проте стають меншими ніж ті які ми занулили. Таким чином, пройшовши матрицю N разів ми можемо отримати діагональну матрицю в результаті, та додаткові матриці множними зліва та справа. На самій матриці ми виділяємо блоки значень 2×2, які захоплюють значення основної діагоналі і діагоналі над або під нею. Оскільки два суміжні блоки мають спільний елемент, ми організовуємо паралельне виконання на відеокарті таким чином, що запускаємо новий потік, коли попередній вже опрацював два блоки. Відповідно, у нас буде $N/2$ потоків які ітераційно опрацьовують матрицю. Розроблений метод містить в собі спосіб оптимізації використання пам'яті за якого зберігаються лише дані основної діагоналі, а також над та під діагоналлю. Окрім цього, запропоновано підхід ефективного використання потоків, де окрім головних потоків, що виконують сам ітераційний процес, в обчисленнях беруть участь допоміжні потоки, які пришвидшують обчислення з множенням матриці Ґівенса на ліві та праві матриці-множники. За результатами цієї роботи також зроблено публікацію [2], де можна дізнатись більше деталей про відповідний алгоритм.

Таблиця 1

Оцінка часу виконання

Розмір матриці	128	256	512	1024	2048
Бідіагоналізація	0.060	0.103	0.216	0.873	6.557
Відхилення (+/- t)	0.005	0.009	0.022	0.009	0.008
Діагоналізація	0.131	0.338	1.280	11.886	64.122
Відхилення (+/- t)	0.016	0.022	0.076	0.802	2.677
Алгоритм	0.191	0.441	1.496	12.759	70.679

Було проведено експерименти на графічному процесорі GA104-300-A1 де генерувались квадратні матриці з випадковими значеннями типу Double діапазоном від 0 до 100. Для кожної згенерованої матриці проводилось по два запуски, оцінювався час виконання кожної частини алгоритму та загальний час, а також брався середній час і діапазон відхилень. В цей час враховується весь процес обчислень, та не враховується процес перевірки похибки обчислень. Час в таблиці 1 наведено в секундах. Експерименти, проведені з поточною версією алгоритму, показали, що розроблений метод бідіагоналізації чудово підходить для виконання на графічному процесорі і демонструє високу швидкість і точність обчислень. Як можна побачити в таблиці 1, діагоналізація займає 90% часу усього алгоритму. Аналіз діагоналізації також допоміг виявити, що хоч групи допоміжних потоків дали хороше пришвидшення, проте цей підхід вносить

проблему дивергенції потоків і складності масштабування на великі матриці. Також виявлено затримки в процесі синхронізації даних між блоками. Наразі ведеться робота над дослідженнями оптимізації діагоналізації в плані обміну інформацією між блоками, а також пришвидшення за допомогою попереднього опрацювання матриці перед виконанням ітераційної частини.

Список джерел

1. Малашонок Г.І., Сухарський С.С., Алгоритм обчислення дводіагональної матриці ортогональним розкладанням на графічному процесорі. Наукові записки НаУКМА. Комп'ютерні науки. 2021. Том 4. doi: 10.18523/2617-3808.2021.4.10-15
2. Сухарський С. С., Алгоритм сингулярного розкладу на графічному процесорі. Проблеми програмування. 2023; 1: 30-37. doi: 10.15407/pp2023.01.030
3. S. Lahabar and P. J. Narayanan, "Singular value decomposition on GPU using CUDA," 2009 IEEE International Symposium on Parallel & Distributed Processing, 2009, pp. 1-10, doi:10.1109/IPDPS.2009.5161058.
4. Jack Dongarra, Mark Gates, Azzam Haidar, Jakub Kurzak, Piotr Luszczek, Stanimire Tomov, and Ichitaro Yamazaki, "The Singular Value Decomposition: Anatomy of Optimizing an Algorithm for Extreme Scale", SIAM Review, v.60, 2018, pp. 808 - 865, doi: 10.1137/17M1117732.
5. Nishida, K., Nakasato, N., & Sedukin, V. "GPU-accelerated randomized singular value decomposition and its applications." Procedia Computer Science, vol. 29, pp. 2109-2119, 2014.

АРХІТЕКТУРА СИСТЕМИ УПРАВЛІННЯ БАЗАМИ ДАНИХ ДЛЯ ГРАНИЧНИХ ОБЧИСЛЕНЬ / DATABASE MANAGEMENT SYSTEM ARCHITECTURE FOR EDGE COMPUTING

Цьоменко Д.М. / Tsomenko D. M.

Київський національний університет імені Тараса Шевченка / Taras Shevchenko National University of Kyiv

Факультет комп'ютерних наук та кібернетики / Faculty of Computer Science and Cybernetics

03022, Україна, м. Київ, пр. Академіка Глушкова, 4д.

E-mail: denistsem@gmail.com, +380954030019

Анотація: This research examines the development of database management systems designed specifically for edge computing environments, incorporating local-first software principles to enhance functionality. The proposed DBMS architecture can process data close to its source by using edge computing, which minimizes latency, reduces the load on the network, and ensures real-time responses for analytics crucial for mission-critical applications. The local-first approach is primarily expressed with Conflict-Free Replicated Data Types, which enable efficient real-time collaboration across distributed environments while maintaining data consistency and resilience. This is especially beneficial for decentralized data processing and localized data storage.

1. Децентралізована обробка даних з використанням граничних обчислень.

З розвитком Інтернету речей та кількості пристроїв в ньому - обсяги даних зростають експоненціально, що призводить до навантаження на мережу та затримках в обробці даних в централізованих хмарних системах. Граничні обчислення вирішують цю задачу дозволяючи обробляти дані ближче до їх джерела зменшуючи затримки та збільшуючи пропускну здатність. Зміщення обробки та аналізу даних ближче до місця збору інформації відкриває можливості для аналізу в реальному часі, що критично для автономних систем, розумних міст, ситуаційної обізнаності на полі бою, авіації, тощо. Наприклад, самокеровані автомобілі потребують швидкої реакції на зібрану інформацію для запобігання зіткненням та адаптації до дорожніх умов. Запропонована архітектура пропонує встановлювати на крайні вузли спеціально розроблену базу даних та залишити більшість обробки на них. Вбудована база даних дозволяє кожному учаснику мережі збирати, обробляти, аналізувати дані незалежно. А принцип local-first дозволяє продовжувати роботу без постійного підключення до хмарного сервісу, таким чином