25. *Rope E.L., Tricoles G., Yue O.C.* Metallic, angular filters for array economy, *in IEEE Antennas Propagat. Int. Symp. Dig*., 1976, pp. 155-157.
26. *Khansen R.* Skaniruyushchie antennye sistemy SVCh [Scanning antenna systems of microwave]. Vol. 1-2. Moscow: Sovetskoe radio, 1966, Vol. 1, 536 p., Vol. 2, 496 p.
27. *Rope E.L., Tricoles G., Yue O-C.* Metallic, Angular Filters For Arrays Economy, *AP-S Session 5, 1610, Tuesday, October 12, Room 163-173*, pp. 155-157.
28. *Kinowski D., Guglielmi M., Roederer A.G.* Angular Bandpass Filters: An Alternative Viewpoint Gives Improved Design Flexibility, *IEEE Trans. Antennas and Propag*., 1995, Vol. AP43, No. 4, pp. 390-395.
29. *Kas'yanov A.O., Obukhovets V.A.* Metallodielektricheskie chastotno-izbiratel'nye poverkhnosti [Metal-dielectric frequency-selective surfaces], *Elektromagnitnye volny i elektronnye sistemy. Radiotekhnika* [Electromagnetic waves and electronic systems. Radio engineering], 2009, Vol. 14, No. 11, pp. 29-38.
30. *Kas'yanov A.O., Kas'yanova A.N.* Elektrodinamicheskiy analiz i razrabotka SAPR-orientirovannykh matematicheskikh modeley pechatnykh antennykh reshetok: monografiya [Electrodynamic analysis and development of CAD-oriented mathematical models of printed antenna arrays: monograph]. Taganrog: Izd-tvo YuFU, 2017, 300 p.

Статью рекомендовал к опубликованию д.т.н., профессор Д.Д. Габриэльян.

Касьянов Александр Олегович - Южный федеральный университет, e-mail: kasyanovao@sfedu.ru; г. Ростов-на-Дону, Россия; кафедра радиотехнических и телекоммуникационных систем, д.т.н., профессор.

Kasyanov Alexander Olegovich – Southern Federal University; e-mail: kasyanovao@sfedu.ru; Rostov-on-Don, Russia; the department of radio engineering and telecommunication systems; dr. of eng. sc.; professor

УДК 004.657 DOI 10.18522/2311-3103-2021-2-104-112

П.А. Курапов

ГИБРИДНОЕ ИСПОЛНЕНИЕ ЗАПРОСОВ К АНАЛИТИЧЕСКИМ БАЗАМ ДАННЫХ

Для повышения эффективности системам исполнения аналитических запросов необходимо использовать все доступные ресурсы современных распределенных гетерогенных систем. Ускорители, сложная иерархия памяти и распределенность вычислений создают возможности для оптимизации производительности. В статье проводится обзор существующих подходов к реализации механизмов исполнения аналитических запросов к СУБД для данных в оперативной памяти с использованием аппаратных ускорителей, в частности, особое внимание уделено графическим ускорителям. За счет массивного параллелизма и высокой пропускной способности памяти устройства графические ускорители представляют перспективную альтернативу основного устройства исполнения аналитических запросов. Существующие методы не задействуют всех возможностей современной аппаратуры и обычно основываются на передаче данных по относительно медленной шине PCIe для исполнения ядер каждого отдельно взятого оператора. Другой проблемой существующих методов является явное разделение кодовой базы кодогенераторов реляционных операторов для ускорителей (графических процессоров) и центрального процессора, и невозможность переиспользования сгенерированного кода для других устройств в системе, что существенно ограничивает возможности их совместного использования с целью повышения производительности. В статье представлен метод эффективного исполнения запросов на примере системы из двух классов устройств (ЦПУ и графический процессор) при помощи компиляции с построением единого, независимого от устройства, промежуточного представления (SPIR-V) и подход к оптимизации соответствующего гибридного физического плана запроса на основе расширенного классического оператора "Exchange" с использованием гетерогенных вычислительных ресурсов и явным контролем уровня паралле-

лизма для каждого устройства. Для поиска оптимального физического плана предложен способ построения модели затрат на основе данных о поведении основных вычислительных паттернов реляционных и вспомогательных операторов. Потенциал прироста производительности за счет оптимизации запросов целиком для наилучшего с точки зрения производительности устройства оценивается с помощью эмпирических данных, полученных для коммерческой СУБД с открытым исходным кодом OmniSci DB. Предварительные результаты демонстрируют возможность ускорения обработки запросов в разы (3-8х) при выборе наиболее подходящего устройства исполнения.

СУБД; графический процессор; оптимизация аналитических запросов; обработка аналитических запросов; гетерогенные вычисления.

P.A. Kurapov

HYBRID EXECUTION OF QUERIES TO ANALYTICAL DATABASES

Analytical database engines should benefit from evolving heterogeneous distributed architectures and utilize their resources efficiently: various accelerators, complex memory hierarchy, and distributed nature of systems bring performance improvement opportunities. The article reviews existing approaches for in-memory DBMS query executor implementation using hardware accelerators, especially GPUs. Massive parallelism and high on-device memory bandwidth make graphics processors a promising alternative as a core query evaluation device. Existing methods do not utilize all modern hardware capabilities and usually are bound, performance-wise, by relatively slow PCIe data transfer in a GPU-as-a-co-processor model for each kernel representing a single relational algebra operator. Another existing approaches' issue is explicit code base separation for relational algebra operators code generation (for CPU and GPU) that significantly limits possibilities of joint device usage for performance increase and make it less feasible. The article presents an efficient query execution method using an example of two device classes (CPU and GPU) by compiling queries into a single, device agnostic, intermediate representation (SPIR-V) and an approach for corresponding hybrid physical query plan optimization based on extended classical "Exchange" operator with explicit control over heterogeneous resources and parallelism level available. A cost model composition process using basic compute DBMS patterns benchmarking and buses bandwidth data for both relational and auxiliary operators is proposed. Potential processing speedup from holistic query optimization is estimated empirically with a commercial open source DBMS OmniSci DB. Preliminary results show significant (3-8x, depending on device choice) possible speedup even with just using the right device for the job.

DBMS; GPU; analytical query optimization; analytical query processing; in-memory databases.

Введение. В последние 10 лет, с развитием GPGPU (General-purpose computing on graphics processing units), СУБД начали активно использовать графические ускорители для обработки аналитических запросов. Многие исследования показали, что за счет массивного параллелизма графических ускорителей, векторных вычислений и высокой пропускной способностью памяти на чипе, выполнение операторов реляционной алгебры на ускорителе показывает значительный прирост в производительности, при условиях, что данные для операции помещаются в память устройства и возможности аппаратуры, такие как SIMD, разделяемая память, группы потоков, используются правильно. Время исполнения оператора реляционной алгебры может быть в разы меньше, чем у соответствующего алгоритма, реализованном на центральном процессоре.

Несмотря на все преимущества, использование графического ускорителя как основного устройства для исполнения запросов не лишено проблем [4]. Доступная память по-прежнему значительно уступает основной в размере. Это ограничение заставляет систему либо хранить данные в основной памяти и каждый раз загружать их в память устройства для обработки, либо организовывать распределенное хранение. И тот, и другой способы опираются на передачу данных через различ-

ные системы соединений, которые значительно уступают в скорости обращения в локальную для устройства память, и потому, становятся узким местом. Другая проблема – зависимость графического процессора от ЦПУ с точки зрения модели исполнения, то есть полагается на внешнюю аллокацию памяти и инициацию вычислений. Наконец, графические процессоры не приспособлены к выполнению кода со сложным ветвлением, интенсивному межпоточному взаимодействию и доступу к сложным структурам данных, что делает выгодным исполнение части операций на других устройствах.

В этой статье мы даем обзор подходов к использованию ускорителей для обработки аналитических запросов к реляционным базам данных, предлагаем архитектуру обработчика аналитических запросов для гетерогенных систем и приводим предварительные эмпирические данные для подтверждения потенциала повышения производительности на примере коммерческой базы данных OmniSci DB [17].

Обзор индустрии. С ростом вычислительной мощности графических процессоров и появлением более удобного способа их программирования (GPGPU), появились базы данных, использующие оптимизированные под архитектуру графических процессоров алгоритмы типичных операторов реляционной алгебры (например BlazingSQL, OmniSciDB, HeteroDB, Kinetica, SQream DB [1]). Для таких систем типичен механизм использования графического процессора для исполнения большинства DML (data manipulation language) запросов на устройстве. Обработка и оптимизация запроса, а также сериализация результатов, обычно, происходит на ЦПУ. Массивный параллелизм графических ускорителей во многих случаях дает значительный прирост производительности в типичных операторах баз данных [20]. Однако не все операции выгодно исполнять на графическом процессоре. Основным ограничением является пропускная способность шины передачи данных (в случае большинства баз данных использующих графический процессор – PCIe). За счет накладных расходов на копирование и использование кода на ЦПУ для запуска ядер для графического процессора (kernel), во многих случаях передавать данные становится неэффективно. Такую проблему можно было бы решить размещением базы данных в локальной памяти устройства, однако ее размер значительно уступает возможностям системной памяти (до 64гб в самых дорогих решениях против >1тб системной памяти), что не позволяет размещать реальные базы данных или делает их размещение слишком дорогим если используется множество графических процессоров.

Другое решение это конвейеризация – объединение нескольких операторов в один kernel с целью сокращения объема передачи данных между хостом и ускорителем. Генерация кода для оператора (цепочки операторов, оканчивающихся pipeline-breaker - оператором, требующим готовности всего результата до исполнения следующего), совместно с пакетной обработкой данных позволяет значительно сократить пересылку данных по медленной PCIe и, тем самым, добиться кратного роста скорости исполнения запроса [2]. Увеличение пропускной способности шины демонстрирует значительное повышение производительности исполнения запроса в целом [3], даже в случае входных данных, не помещающихся в память целиком (morsel-driven).

Использование графических процессоров для ускорения работы базы данных не ограничивается только исполнением отдельных операторов, например, исследовались подходы совместного использования с ЦПУ. Один из подходов [5] решает проблему высокой нагрузки на шину PCIe и недостаточности размера «быстрой» памяти графического процессора с помощью побитового разделения данных – горячие части данных (над которыми производятся вычислительно-емкие и независимые операции) хранятся в локальной памяти устройства, менее активно ис-

пользуемые – в системной памяти. Каждый запрос обрабатывается в 2 стадии: сначала неточная фильтрация записей, затем промежуточный результат передается на ЦПУ и объединяется с остаточными данными в основной памяти. Другой способ совместного использования ЦПУ и графического ускорителя – использование ускорителя для поиска оптимального плана исполнения запроса, а точнее, для оценки селективности с целью повышения качества физического плана [7].

Эффективная координация ЦПУ и графического процессора (и других ускорителей, таких как FPGA) имеет большой потенциал ([6–9]) повышения производительности систем, поэтому интерес к гибридному исполнению запросов растет. Одна из первых работ в этой области [10] показала 42% прироста производительности отдельно взятого простого запроса с помощью выбора подходящего, с точки зрения решаемой задачи, устройства.

С добавлением устройств и уровней иерархии памяти в систему, растет пространство перебора возможных вариантов исполнения запроса, и существующие техники оптимизации планов становятся менее эффективны, поэтому оптимизаторы гибридных запросов оказываются далеки от генерации оптимальных планов. Наиболее распространенным способом использования гетерогенных систем для обработки запросов является пооператорное планирование [11, 12] – для отдельно взятого оператора оценивается стоимость исполнения на ЦПУ и сравнивается со стоимостью передачи данных (туда и обратно) и время исполнения на ускорителе.

Например, Hawk [11] решает задачу гибридных запросов следующим образом. Реляционные операторы транслируются во внутреннее представление (pipeline programs), которое строится по принципам максимизации локальности по данным и коду. Последовательность реляционных операторов, не требующих полной материализации результатов (то есть, можно оставлять данные в регистрах), можно объединять и исполнять совместно [13]. Каждая «программа» выражается в виде специального оператора с платформо-зависимыми параметрами – модификации (варьируется способ доступа к памяти, стратегия исполнения, количество потоков), которые может производить оптимизатор для генерации планов специфичных какому-либо устройству. Планы оцениваются с помощью обучаемой модели затрат, и наилучший по стоимости выбирается в качестве оптимального. Проблема такого подхода заключается в том, что оптимизация локальна по отношению к оператору и не учитывает возможное ускорение цепочки операторов при отсутствии перемещения данных. К тому же, поскольку генератор кода Hawk выдает исходный OpenCL код, возникают дополнительные расходы на его компиляцию.

Избежать накладных расходов на генерацию исходного кода позволит введение «независимого» от устройства промежуточного представления. Помимо упрощения логики компиляции, промежуточное представление может абстрагировать архитектурные и микроархитектурные свойства аппаратуры. Так, [14] решает проблему адаптации обработчика для MonetDB с сохранением конкурентной производительности с помощью введения промежуточной алгебры, параметризованной уровнем параллелизма. Сама по себе алгебра поддерживает стандартные оптимизации, такие как смена размещения памяти (row/column store), различные стратегии материализации результатов, совместное использование общих результатов, векторизацию, слияние и разделение циклов. Как и в случае Hawk, генерацией кода и его исполнением занимается OpenCL. Операторы промежуточной алгебры разделяются по уровню параллелизма на «фрагменты» кода (термин «фрагмент» заимствован из [13] – работа предлагает реализовать операторы реляционной алгебры через примитивы промежуточного представления компилятора и передавать результаты вычис-

лений напрямую по графу потребителю). Каждый фрагмент может быть скомпилирован в отдельный kernel и исполнен с максимально возможной эффективностью с точки зрения использования внутреннего параллелизма.

Использование ускорителя для исполнения оператора или запроса целиком оставляет ЦПУ незадействованным, что в случае серверной аппаратуры – значительная потеря вычислительной мощности. HetExchange [15] решает эту проблему введением дополнительных операторов физического плана, управляющих передачей данных с ЦПУ на графический процессор и обратно (device-crossing), разделением работы (оператор router – отвечает за уровень гетерогенного параллелизма), и контролем назначения работы нужному узлу (совместно с data-flow операторами, перемещающими гетерогенную память).

Материалы и методы. Параллельное исполнение операторов на разных устройствах требует соответствующих реализаций алгоритмов. С точки зрений разработки и поддержки системы наиболее эффективный путь – использование существующего стека технологий, такого как OpenCL для генерации кода под конкретные платформы. Например, CoGaDB генерирует исходных код на OpenCL C для запроса по статически заданным шаблонам. Такой подход использует преимущество существующих генераторов кода для устройств (ЦПУ, графический процессор), однако вносит дополнительные затраты на синтаксический и семантический анализ входного языка. В нашем исследовании мы используем более эффективный метод: сочетание использования гетерогенного плана запроса и прямой генерации промежуточного представления компилятора из абстрактного дерева операторов. Proteus (HAPE) [15] расширяет классический Exchange оператор, вводя дополнительные операторы передачи данных и контроля потока исполнения для утилизации гетерогенности и параллелизма вычислителей (рис. 1).

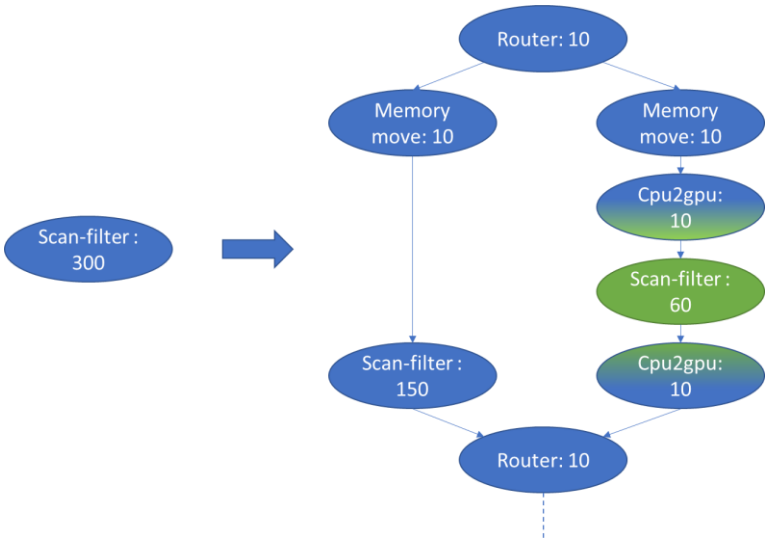

Рис. 1. Пример преобразования физического плана для ЦПУ в гетерогенный. Цветом отмечена смена устройства исполнения. Цифры – стоимость оператора в условных единицах времени исполнения

Мы исследуем возможность использования гетерогенного плана для создания промежуточного представления SPIR-V (Standard Portable Intermediate Representation) [18], которое не зависит от устройства и, как в случае с OpenCL, может выполняться на любом процессоре, для которого существует генератор ко-

да. Часть специфичной функциональности для устройства (к примеру, использование быстрой разделяемой памяти для графического ускорителя) компилируется заранее и соединяется на этапе генерации кода с запросом.

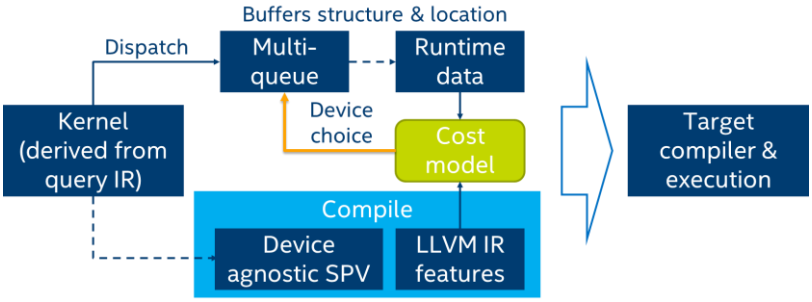

Рис. 2. Принципиальная архитектура исполнения сгенерированного кода на произвольной доступной платформе

В этой статье мы делаем первый шаг на пути к описанной схеме и предлагаем архитектуру генерации устройство-независимого представления (рис. 2), а также делаем предварительные замеры на коммерческой базе данных с открытым исходным кодом OmniSci DB с целью показать потенциальный выигрыш в производительности от использования гетерогенности. Для входного запроса строится абстрактное синтаксическое дерево (план), в узлах которого хранится информация о реляционных операторах. Исполнение запроса начинается с генерации промежуточного представления (LLVM IR) компилятора из плана запроса. К сгенерированному IR привязываются статически скомпилированные (до промежуточного представления) функции, реализующие стандартные алгоритмы (такие как hash-join), применяются классические оптимизации и, наконец, модуль транслируется в переносимое представление (SPIR-V). На этом этапе, с небольшими оговорками, полученный модуль может выполняться на любом доступном в системе устройстве.

Результаты. HAPE показывает значительный прирост производительности в случае использования гетерогенных планов исполнения запросов (до 3.5 раз). Здесь мы приводим эмпирическую оценку потенциального выигрыша в производительность от использования подходящего устройства для запроса, когда оптимизация происходит на уровне всего запроса, а не отдельных операторов.

OmniSci DB позволяет выполнять Just-In-Time компиляцию запроса для ЦПУ и графического процессора. На графике (рис. 3) представлены результаты замеров времени исполнения 4х запросов к данным NYC Taxi Benchmark [19], исполненных на Intel(R) Core(TM) i7-6950X CPU @ 3.00GHz и Nvidia GeForce GTX 1080 GPU. Исходный код SQL для запросов Q1-Q4 находится в приложении. Цифры отражают время исполнения сгенерированного кода в миллисекундах с накладными расходами на вызов процедур графического процессора и передачей данных, если это необходимо. Время компиляции не учитывается. Полученные значения – геометрическое среднее для 9 измерений, 2 из которых отбрасываются (максимум и минимум). Как показывает график (рис. 3), несмотря на то, что СУБД оптимизирована для исполнения запросов на графическом процессоре, некоторые запросы (Q2), выгоднее обрабатывать на ЦПУ, если его оптимизация производится на уровне всего запроса.

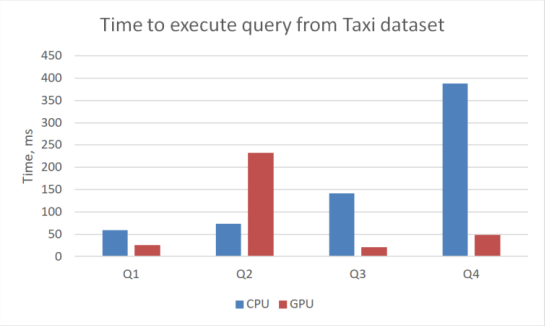

Рис. 3. График зависимости времени исполнения запросов к OmniSci DB в режимах ЦПУ и GPU для датасета NYC Taxi

Обсуждение. Предварительные результаты показывают возможность получения значительного (3–8 раз, в зависимости от устройств) прироста производительности, даже при использовании только подходящего для запроса устройства. Предложенная архитектура обработчика запросов позволяет абстрагировать имплементации операторов для различных устройств с помощью существующих стеков компиляции и генерации кода для различных платформ.

С увеличением количества операторов и контекстов их использования расширяется пространство перебора возможных физических планов исполнения запроса. Модель затрат для операторов физического плана позволит исследовать пространство поиска оптимального (достаточно эффективного) решения. ProGraML [16] показывает, как с помощью извлечения свойств промежуточного представления получить модель выбора наиболее эффективного устройства исполнения. В случае аналитических запросов задача упрощается, поскольку основные паттерны исполнения известны заранее – каждый оператор соответствует некоторому паттерну с точки зрения исполнения и профилю обращения в память. На основе данных о поведении таких паттернов можно построить модель затрат. На этапе оптимизации обработчику будет необходимо вычислить оценку стоимости вычисления оператора на конкретном устройстве, стоимость перемещения данных, которая зависит от скорости шины, и учесть накладные расходы на использование операторов, поддерживающих гетерогенное исполнение. Поскольку в нашем подходе генерируемый код практически не меняется в зависимости от конечного устройства, обучаемая модель, опирающаяся на характеристики промежуточного представления и на данные о загруженности устройства и их размере, будет устойчива к изменениям.

Заключение. В статье рассмотрена проблема оптимизации выполнения аналитических запросов для гетерогенных систем, рассмотрены существующие подходы к использованию аппаратных акселераторов и освещены проблемы этих методов. Для решения задачи совместной утилизации ресурсов предложена архитектура обработчика запросов, которая позволяет минимизировать затраты на поддержку и разработку системы за счет унификации кодогенерации, в отличие от существующих решений, без ущерба производительности, как, например, в случае генерации исходного кода OpenCL в качестве промежуточного представления. Предложенная архитектура в том числе позволяет проще расширять список поддерживаемых аппаратных ускорителей и дает возможность реализации единой модели затрат. В статье так же предложена схема построения такой модели на основе основных вычислительных паттернов. Приводятся эмпирические подтверждения потенциального повышения производительности коммерческой системы за счет использования гете-

рогенности. Следующим шагом нашего исследования станет построение модели оптимизации гетерогенного плана запроса на основе данных об относительной стоимости исполнения операторов на различных устройствах.

Приложение

Q1:

```
SELECT cab_type,
    count(*)
FROM trips
GROUP BY cab_type;
```

Q2:

```
SELECT passenger_count,
    avg(total_amount)
FROM trips
GROUP BY passenger_count;
```

Q3:

```
SELECT passenger_count,
    extract(year from pickup_datetime) AS pickup_year,
    count(*)
FROM trips
GROUP BY passenger_count,
      pickup_year;
```

Q4:

```
SELECT passenger_count,
    extract(year from pickup_datetime) AS pickup_year,
    cast(trip_distance as int) AS distance,
    count(*) AS the_count
FROM trips
GROUP BY passenger_count,
      pickup_year,
      distance
ORDER BY pickup_year,
   the_count desc;
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Hawon Chu, Seounghyun Kim, Joo-Young Lee, and Young-Kyoon Suh.* Empirical evaluation across multiple GPU-accelerated DBMSes, *In Proceedings of the 16th International Workshop on Data Management on New Hardware* (*DaMoN '20*). Association for Computing Machinery, New York, NY, USA, 2020, Article 16, pp. 1–3. Doi: https://doi.org/10.1145/3399666.3399907.
2. *Henning Funke, Sebastian Breß, Stefan Noll, Volker Markl, and Jens Teubner*. Pipelined Query Processing in Coprocessor Environments. In *Proceedings of the 2018 International Conference on Management of Data* (*SIGMOD '18*). Association for Computing Machinery, New York, NY, USA, 2018, 16031618. Doi: https://doi.org/10.1145/3183713.3183734.
3. Clemens Lutz, Sebastian Breß, Steffen Zeuch, Tilmann Rabl, and Volker Markl. 2020. Pump Up the Volume: Processing Large Data on GPUs with Fast Interconnects. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (*SIGMOD '20*). Association for Computing Machinery, New York, NY, USA, pp. 1633-1649. Doi:https://doi.org/10.1145/3318464.3389705.
4. *Breß S., Heimel M., Siegmund N., Bellatreche L., Saake G.* GPU-Accelerated Database Systems: Survey and Open Challenges. In: Hameurlain A. et al. (eds) Transactions on Large-Scale Data- and Knowledge-Centered Systems XV. Lecture Notes in Computer Science, Vol. 8920. Springer, Berlin, Heidelberg, 2014. Doi: https://doi.org/10.1007/978-3-662-45761-0_1.

5. *Breß S., Heimel M., Siegmund N., Bellatreche L., Saake G.* Exploring the Design Space of a GPU-Aware Database Architecture. In: Catania B. et al. (eds) New Trends in Databases and Information Systems. Advances in Intelligent Systems and Computing. Springer, Cham, 2014, Vol. 241. Doi: https://doi.org/10.1007/978-3-319-01863-8_25.
6. *Bingsheng He, Mian Lu, Ke Yang, Rui Fang, Naga K. Govindaraju, Qiong Luo, and Pedro V. Sander.* Relational query coprocessing on graphics processors. ACM Trans. Database Syst. 34, 4, Article 21 (December 2009), 39 p. Doi: https://doi.org/10.1145/1620585.1620588.
7. *Heimel M., Markl V.* A First Step Towards GPU-assisted Query Optimization. In: Bordawekar, R. and Lang, C.A. (eds.) ADMS@VLDB, 2012, pp. 33-44.
8. Peter Bakkum and Kevin Skadron. Accelerating SQL database operations on a GPU with CUDA. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units* (*GPGPU-3*). Association for Computing Machinery, New York, NY, USA, 2010. pp. 94-103. Doi: https://doi.org/10.1145/1735688.1735706.
9. *Johns Paul, Jiong He, and Bingsheng He.* GPL: A GPU-based Pipelined Query Processing Engine. In *Proceedings of the 2016 International Conference on Management of Data* (*SIGMOD '16*). Association for Computing Machinery, New York, NY, USA, 2016, pp. 1935-1950. Doi: https://doi.org/10.1145/2882903.2915224.
10. *Ilic, Aleksandar & Pratas, Frederico & Trancoso, Pedro & Sousa, Leonel.* High-Performance Computing on Heterogeneous Systems: Database Queries on ЦПУ and GPU, 2011. Doi 10.3233/978-1-60750-803-8-202.
11. *Sebastian Breß, Bastian Köcher, Henning Funke, Steffen Zeuch, Tilmann Rabl, and Volker Markl.* Generating custom code for efficient query execution on heterogeneous processors, *The VLDB Journal* 27, 6 (December 2018), pp. 797-822. Doi: https://doi.org/10.1007/s00778-018-0512-y.
12. *Owaida M., Sidler D., Kara K. and Alonso G.* Centaur: A Framework for Hybrid ЦПУ-FPGA Databases, *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2017*, pp. 211-218. Doi: 10.1109/FCCM.2017.37.
13. *Thomas Neumann.* Efficiently compiling efficient query plans for modern hardware. *Proc. VLDB Endow.* 4, 9 (June 2011), pp. 539-550. DOI:https://doi.org/10.14778/2002938.2002940.
14. *Holger Pirk, Oscar Moll, Matei Zaharia, and Sam Madden*. Voodoo - a vector algebra for portable database performance on modern hardware. *Proc. VLDB Endow.* 9, 14 (October 2016), 1707–1718. Doi: https://doi.org/10.14778/3007328.3007336.
15. *Periklis Chrysogelos, Manos Karpathiotakis, Raja Appuswamy, and Anastasia Ailamaki.* HetExchange: encapsulating heterogeneous ЦПУ-GPU parallelism in JIT compiled engines. *Proc. VLDB Endow.* 12, 5 (January 2019), pp. 544-556. Doi: https://doi.org/10.14778/3303753.3303760.
16. *Cummins C., Fisches Z.V., Ben-Nun T., Hoefler T., and Leather H*. Programl: Graph-based deep learning for program optimization and analysis, *arXiv preprint arXiv:2003.10536*, 2020.
17. *Christopher Root and Todd Mostak*. MapD: a GPU-powered big data analytics and visualization platform. In *ACM SIGGRAPH 2016 Talks* (*SIGGRAPH '16*). Association for Computing Machinery, New York, NY, USA, 2016, Article 73, pp. 1-2. Doi: https://doi.org/10.1145/2897839.2927468.
18. https://www.khronos.org/registry/spir-v/.
19. New York (N.Y.). Taxi and Limousine Commission. New York City Taxi Trip Data, 2009-2018. Inter-university Consortium for Political and Social Research [distributor], 2019-02-20. https://doi.org/10.3886/ICPSR37254.v1.
20. *Shanbhag, Anil and Madden, Samuel and Yu, Xiangyao*. A Study of theFundamental Performance Characteristics of GPUs and CPUs for DatabaseAnalytics, *Proceedings of the 2020 ACM SIGMOD International Conferenceon Management of Data, Association for Computing Machinery, New York,NY, USA, 2020*, pp. 1617-1632.

Статью рекомендовал к опубликованию д.т.н., профессор В.М. Курейчик.

Курапов Петр Александрович – МФТИ; e-mail: petr.kurapov@gmail.com; Москва, Россия; кафедра микропроцессорных технологий в интеллектуальных системах управления; аспирант.

Kurapov Petr Alexandrovich – Moscow Institute of Physics and Technology; e-mail: petr.kurapov@gmail.com; Moscow, Russia; the department of microprocessor technologies in intelligent control systems; graduate student.