Санкт-Петербургский Государственный Университет

Программная инженерия
Кафедра системного программирования

Орачев Егор Станиславович

Реализация алгоритма поиска путей в графовых базах данных через тензорное произведение на GPGPU

Бакалаврская работа

Научный руководитель:
к. ф.-м. н., доцент С. В. Григорьев

Рецензент:

Санкт-Петербург
2020

Оглавление

Введение

Все чаще современные системы аналитики и рекомендаций строятся на основе анализа данных, структурированных с использованием *графовой модели*. В данной модели основные сущности представляются вершинами графа, а отношения между сущностями — ориентированными ребрами с различными метками. Подобная модель позволяет относительно легко и практически в явном виде моделировать сложные иерархические структуры, которые не так просто представить, например, в классической *реляционной модели*. В качестве основных областей применения графовой модели можно выделить следующие: графовые базы данных [4], анализ RDF данных [5], биоинформатика [15] и статический анализ кода [10].

Поскольку графовая модель используется для моделирования отношений между объектами, при решении прикладных задач возникает необходимость выявления более сложных взаимоотношений между объектами. Для этого чаще всего формируются запросы в специализированных программных средствах для управления графовыми базами данных. В качестве запроса можно использовать некоторый *шаблон* на путь в графе, который будет связывать объекты, т.е. выражать взаимосвязь между ними. В качестве такого шаблона можно использовать формальные грамматики, например, регулярные или контекстно-свободные (КС). Используя вычислительно более выразительные грамматики, можно формировать более сложные запросы и выявлять нестандартные и скрытые ранее взаимоотношения между объектами. Например, *same-generation queries* [1], сходные с сбалансированными скобочными последовательностями Дика, могут быть выражены КС грамматиками, в отличие от регулярных.

Результатом запроса может быть множество пар объектов, между которыми существует путь в графе, удовлетворяющий заданным ограничениям. Также может возвращаться один экземпляр такого пути для каждой пары объектов или итератор всех путей, что зависит от семантики запроса. Поскольку один и тот же запрос может иметь разную семантику, требуются различные программные и алгоритмические средства для его выполнения.

Запросы с регулярными ограничениями изучены достаточно хорошо, языковая и программная поддержка выполнения подобных запросов присутствует в некоторых в современных графовых базах данных. Однако, полноценная поддержка запросов с КС ограничениями до сих пор не представлена. Существуют алгоритмы [5, 11, 3, 6, 13] для вычисления запросов с КС ограничениями, но потребуется еще время, прежде чем появиться полноценная высокпроизводительная реализация одного из алгоритмов, способная обрабатывать реальные графовые данные.

Работы [8, 7] в качестве реализации алгоритма [3] показывают, что возможно использовать GPGPU для выполнения наиболее вычислительно сложных частей алгоритма, что дает *существенный* прирост в производительности. Недавно представленный алгоритм [6] для вычисления запросов с КС ограничениями полагается на операции линейной алгебры: произведение Кронекера (частный случай тензорного произведения), умножение и сложение матриц в полукольце булевой алгебры. Данный алгоритм в сравнении с [3] позволяет выполнять запросы для всех ранее упомянутых семантик, потенциально поддерживает бóльшие по размеру КС запросы, с незначительными накладными расходами позволяет выполнять запросы с регулярными ограничениями, а также хорошо реализуется с помощью программных средств для вычисления на GPGPU.

Таким образом, важной задачей является реализация и апробация перспективного алгоритма [6] для выполнения запросов с КС и регулярными ограничениям, а также разработка программной библиотеки для работы с примитивами линейной булевой алгебры, которая позволила бы упростить прототипирование и реализацию подобного и будущих алгоритмов на GPGPU, в частности, на платформе NVIDIA CUDA [14].

1. Постановка задачи

Цель данной работы — реализация алгоритма поиска путей в графовых базах данных через тензорное произведение на платформе NVIDIA CUDA в качестве GPGPU технологии. Для ее достижения были поставлены следующие задачи:

- Реализация библиотеки для работы с примитивами булевой алгебры на GPGPU
- Реализация интерфейса для работы с примитивами библиотеки в тестовой инфраструктуре
- Реализация алгоритма поиска путей с КС ограничениями
- Апробация алгоритма с использованием синтетических и реальных данных

2. Обзор предметной области

2.1. Терминология

В этой секции изложены основные определения и факты из теории графов и формальных языков, необходимые для понимания предметной области.

Ориентированный граф с метками $\mathcal{G} = (V, E, L)$ это тройка объектов, где V конечное непустое множество вершин графа, $E \subseteq V \times L \times V$ конечное множество ребер графа, L конечное множество меток графа. Здесь и далее будем считать, что вершины графа индексируются целыми числами, т.е. $V = \{0 \ ... \ |V| - 1\}$.

Граф $\mathcal{G} = (V, E, L)$ можно представить в виде матрицы смежности M размером $|V| \times |V|$, где $M[i, j] = \{l \mid (i, l, j) \in E\}$. Используя булеву матричную декомпозицию, можно представить матрицу смежности в виде набора матриц $\mathcal{M} = \{M^l \mid l \in L, M^l[i, j] = 1 \iff l \in M[i, l]\}$.

Путь π в графе $\mathcal{G} = (V, E, L)$ это последовательность ребер e_0, e_1, e_{n-1}, где $e_i = (v_i, l_i, u_i) \in E$ и для любых $e_i, e_{i+1} : u_i = v_{i+1}$. Путь между вершинами v и u будем обозначать как $v\pi u$. Слово, которое формирует путь $\pi = (v_0, l_0, v_1), ..., (v_{n-1}, l_{n-1}, v_n)$ будем обозначать как $\omega(\pi) = l_0 ... l_{n-1}$, что является конкатенацией меток вдоль этого пути π.

Контекстно-свободная грамматика $G = (\Sigma, N, P, S)$ это четверка объектов, где Σ конечное множестве терминалов или алфавит, N конечное множество нетерминалов, P конечное множество правил вывода вида $A \to \gamma, \gamma \in (N \cup \Sigma)^*$, $S \in N$ стартовый нетерминал.

Язык L над конечным алфавитом символов Σ это множество всевозможных слов, составленных из символов этого алфавита, т.е. $L = \{\omega \mid w \in \Sigma^*\}$.

2.2. Поиск путей с ограничениями

При вычислении запроса p на поиск путей в графе $\mathcal{G} = (V, E, L)$ в качестве ограничения выступает некоторый язык L, которому должны удовлетворять результирующие пути.

Поиск путей в графе с семантикой **достижимости**, это поиск всех таких пар вершин (v, u), что между ними существует путь $v\pi u$, такой что $\omega(\pi) \in L$. Результат запроса обозначается как $R = \{(v, u) \mid \exists v\pi u : \omega(\pi) \in L\}$.

Поиск путей в графе с семантикой **всех путей**, это поиск всех таких путей $v\pi u$, что $\omega(\pi) \in L$. Результат запроса обозначается как $\Pi = \{v\pi u \mid v\pi u : \omega(\pi) \in L\}$.

Необходимо отметить, что множество Π может быть бесконечным, поэтому в качестве результата запроса предполагается не всё множество в явном виде, а некоторый *итератор* или алгоритм, который позволит последовательно извлекать все пути.

Семантика **одного пути** является ослабленной формулировкой семантики всех путей, так как для получения результата достаточно найти всего один путь вида $v\pi u : \omega(\pi) \in L$ для каждой пары $(v, u) \in R$.

Поскольку язык L может быть бесконечным, при составлении запросов используют не множество L в явном виде, а некоторое правило формирования слов этого языка. В качестве таких правил и выступают регулярные выражения или КС грамматики. При именовании запросов отталкиваются от типа правил, поэтому запросы именуются как регулярные или КС соответственно.

2.3. Существующие решения

Впервые проблема выполнения запросов с контекстно-свободными ограничениями была сформулирована в 1990 году в работе МихалисаЯннакакиса [17]. С того времени были представлены многие работы, в которых так или иначе предлагалось решение данной проблемы. Однако в недавнем исследовании Йохем Куиджперс и др. [9] на основе сравнения нескольких алгоритмов [11, 3, 16] для выполнения запросов с контекстно-свободными ограничениями заключили, что существующие алгоритмы неприменимы для анализа реальных данных в силу того, что обработка таких данных занимает значительное время. Стоит отметить, что алгоритмы, используемые в статье, были реализованы на языке программирования *Java* и исполнялись в среде *JVM* в однопоточном режиме, что не является сколь-угодно производительным решением.

Это подтверждают результаты работы [7], в которой с использование программных и аппаратных средств NVIDIA CUDA был реализован алгоритм Рустама Азимова [3]. В данном алгоритме задача поиска путей с КС ограничениями для семантики одного пути сведена к операциям линейной алгебры, что позволяет использовать высокопроизводительные библиотеки для выполнения данных операций на GPGPU.

2.4. Поиск путей через произведение Кронекера

Недавно представленный алгоритм [6] для выполнения КС запросов использует подобную технику сведения вычислений к операциям булевой алгебры: произведению Кронекера, матричному умножению и сложению. Однако структура алгоритма такова, что он позволяет выполнять запросы сразу в семантике достижимости и семантике всех путей, способен работать с КС грамматиками существенно большего размера, также имеет относительно небольшие накладные расходы при вычислении запросов с регулярными ограничениями, что делает его потенциально применимым для решения этого класса проблем.

Прежде чем рассмотреть алгоритм поиска путей через произведение Кронекера, еще раз обратимся к теории формальных языков и некоторым элементам линейной алгебры, чтобы выработать на интуитивном уровне понимание идеи алгоритма.

Детерминированный конечный автомат (ДКА) $F = (\Sigma, Q, Q_s, Q_f, \delta)$ это пятерка объектов, где Σ конечное множество входных символов или алфавит, Q конечное множество состояний, $Q_s \subseteq Q$ множество стартовых состояний, $Q_f \subseteq Q$ множество конечных состояний, $\delta : \Sigma \times Q \to Q$ функция переходов автомата. Любое регулярное выражение может быть преобразовано в соответствующий ДКА без ε-переходов [12].

Рекурсивный автомат (РА) $R = (M, m, \{C_i\}_{i \in M})$ это тройка объектов, где M конечное множество меток компонентных ДКА, называемых далее *модули*, m метка стартового модуля, $\{C_i\}$ множество модулей, где модуль $C_i = (\Sigma \cup M, Q_i, q_i^0, F_i, \delta_i)$: состоит из:

- $\Sigma \cup M$ множество символов модуля, $\Sigma \cap M = \emptyset$
- Q_i конечное множество состояний модуля, $Q_i \cap Q_j = \emptyset, \forall i \neq j$
- q_i^0 стартовое состояние модуля
- $F_i \subseteq Q_i$ множество конечных состояний модуля
- $\delta_i : Q_i \times (\Sigma \cup M) \to Q_i$ функция переходов

Рекурсивный автомат ведет себя как набор ДКА или модулей [2]. Эти модули очень сходны с ДКА при обработке входных последовательностей символов, однако модули РА способны обрабатывать дополнительные *рекурсивные вызовы* за счет неявного *стека вызовов*, который присутствует во время работы РА. С точки зрения прикладного программиста это похоже на рекурсивные вызовы одних функций из других с той разницей, что вместо функций здесь выступают модули РА.

Рекурсивные автоматы по своей вычислительной мощности эквивалентны автоматам на основе стека [2]. А поскольку подобный стековый автомат способен распознавать КС грамматику [12], рекурсивные автоматы эквивалентны КС грамматикам. Это позволяет корректно использовать РА для кодирования входной КС грамматики запроса.

Список литературы

[1] Abiteboul Serge, Hull Richard, Vianu Victor. Foundations of Databases. — 1995. — 01. — ISBN: 0-201-53771-0.

[2] Analysis of Recursive State Machines / Rajeev Alur, Michael Benedikt, Kousha Etessami et al. // ACM Trans. Program. Lang. Syst. — 2005. — Jul. — Vol. 27, no. 4. — P. 786–818. — Access mode: https://doi.org/10.1145/1075382.1075387.

[3] Azimov Rustam, Grigorev Semyon. Context-free path querying by matrix multiplication. — 2018. — 06. — P. 1–10.

[4] Barceló Baeza Pablo. Querying Graph Databases // Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems. — PODS '13. — New York, NY, USA : Association for Computing Machinery, 2013. — P. 175–188. — Access mode: https://doi.org/10.1145/2463664.2465216.

[5] Context-Free Path Queries on RDF Graphs / Xiaowang Zhang, Zhiyong Feng, Xin Wang et al. // CoRR. — 2015. — Vol. abs/1506.00743. — 1506.00743.

[6] Context-Free Path Querying by Kronecker Product / Egor Orachev, Ilya Epelbaum, Rustam Azimov, Semyon Grigorev. — 2020. — 08. — P. 49–59. — ISBN: 978-3-030-54831-5.

[7] Context-Free Path Querying with Single-Path Semantics by Matrix Multiplication / Arseniy Terekhov, Artyom Khoroshev, Rustam Azimov, Semyon Grigorev. — 2020. — 06. — P. 1–12.

[8] Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication / Nikita Mishin, Iaroslav Sokolov, Egor Spirin et al. — 2019. — 06. — P. 1–5.

[9] An Experimental Study of Context-Free Path Query Evaluation Methods / Jochem Kuijpers, George Fletcher, Nikolay Yakovets,

Tobias Lindaaker // Proceedings of the 31st International Conference on Scientific and Statistical Database Management. — SSDBM '19. — New York, NY, USA : ACM, 2019. — P. 121–132. — Access mode: http://doi.acm.org/10.1145/3335783.3335791.

[10] Fast Algorithms for Dyck-CFL-Reachability with Applications to Alias Analysis / Qirun Zhang, Michael R. Lyu, Hao Yuan, Zhendong Su // SIGPLAN Not. — 2013. — Jun. — Vol. 48, no. 6. — P. 435–446. — Access mode: https://doi.org/10.1145/2499370.2462159.

[11] Hellings Jelle. Path Results for Context-free Grammar Queries on Graphs. — 2015. — 02.

[12] Hopcroft John E., Motwani Rajeev, Ullman Jeffrey D. Introduction to Automata Theory, Languages, and Computation (3rd Edition). — USA : Addison-Wesley Longman Publishing Co., Inc., 2006. — ISBN: 0321455363.

[13] Medeiros Ciro, Musicante Martin, Costa Umberto. An Algorithm for Context-Free Path Queries over Graph Databases. — 2020. — 04.

[14] NVIDIA. CUDA Toolkit Documentation // NVIDIA Developer Zone. — 2020. — Access mode: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (online; accessed: 01.12.2020).

[15] Quantifying variances in comparative RNA secondary structure prediction / James Anderson, Adám Novák, Zsuzsanna Sükösd et al. // BMC bioinformatics. — 2013. — 05. — Vol. 14. — P. 149.

[16] Santos Fred, Costa Umberto, Musicante Martin. A Bottom-Up Algorithm for Answering Context-Free Path Queries in Graph Databases. — 2018. — 01. — P. 225–233. — ISBN: 978-3-319-91661-3.

[17] Yannakakis Mihalis. Graph-Theoretic Methods in Database Theory // Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. —

PODS '90. — New York, NY, USA : Association for Computing Machinery, 1990. — P. 230–242. — Access mode: https://doi.org/10.1145/298514.298576.