На правах рукописи

Снытникова Татьяна Валентиновна

Эффективная реализация модели ассоциативных вычислений на графических ускорителях для решения задач на графах

2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени

кандидата технических наук

Новосибирск – 2022

Работа выполнена в Федеральном государственном бюджетном учреждении науки Институте вычислительной математики и математической геофизики Сибирского отделения Российской академии наук

Научный руководитель:	**Глинский Борис Михайлович**, доктор технических наук
Официальные оппоненты:	**Абрамов Сергей Михайлович**, член-корреспондент РАН, доктор физико-математических наук, профессор, Федеральное государственное бюджетное учреждение науки «Институт программных систем имени А.К. Айламазяна» Российской академии наук директор ИЦМС ИПС им. А.К.Айламазяна РАН **Курносов Михаил Георгиевич**, доктор технических наук, профессор, Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики», директор Центра параллельных вычислительных технологий
Ведущая организация:	Федеральное государственное бюджетное учреждение науки «Институт автоматики и электрометрии» Сибирского отделения Российской академии наук

Защита состоится 17 января 2023 г. в 15:00 часов на заседании диссертационного совета 24.1.047.01 на базе Федерального государственного бюджетного учреждения науки Института вычислительной математики и математической геофизики Сибирского отделения Российской академии наук (ИВМиМГ СО РАН) по адресу: 630090, г. Новосибирск, пр. Академика Лаврентьева, 6, конференц-зал ИВМиМГ СО РАН, тел. (383) 330-71-59.

С диссертацией можно ознакомиться в библиотеке и на сайте ИВМиМГ СО РАН, http://icmmg.nsc.ru.

Автореферат разослан 7 ноября 2022 г.

Ученый секретарь
диссертационного совета 24.1.047.01,
доктор физико-математических наук  Сорокин Сергей Борисович

Общая характеристика работы

Актуальность работы

По статистике в современном мире объем цифровой информации удваивается каждые восемнадцать месяцев. Большая часть информации не является структурированной (организованной в базы данных и знаний). Поэтому обеспечение быстрого поиска образца среди больших объемов информации является актуальной проблемой современной информационной науки. Это привело к выделению науки о данных, как отдельной области науки. При этом известно, что алгоритмы поиска по несортированным данным являются узким местом для вычислительных машин фон-неймановского типа.

В то же время ассоциативные параллельные модели и архитектуры обладают следующим свойством: время выполнения базовых операции поиска ($=$,$<$, $>$, min, max) в массиве не зависит от числа строк. Поэтому все ассоциативные архитектуры разрабатывались и создавались для выполнения конкретных задач, в которых критичны лимиты времени поиска по большому массиву неструктурированных данных.

Развитие ассоциативных моделей остается актуальным. Ведется работа над реализацией ассоциативных моделей на существующем оборудовании. Также создается новое аппаратное обеспечение для ассоциативных параллельных вычислений (чипы ассоциативной памяти, ассоциативные процессоры и системы). Для этих моделей и систем разрабатываются алгоритмы. В качестве ассоциативной модели ранее Непомнящей Анной Шмилевной была предложена STAR-машина. Таким образом актуальность работы заключается в предоставлении возможности использовать преимущества ассоциативных вычислений на графических ускорителях.

Степень разработанности темы

С одной стороны, реализация ассоциативных процессоров на системах неассоциативного типа встречаются достаточно регулярно. Но обычно речь

идет об экспериментальных образцах ассоциативных процессоров. Целью таких реализаций является предсказание свойств этих процессоров.При этом моделируются только низкоуровневые операции.

С другой стороны, параллельно с конструированием новых ассоциативных систем развиваются две модели ассоциативных вычислений. Обе абстрактные модели основаны на системе STARAN и используют языки высокого уровня: STAR-машина[1] и ASC[2]. Для абстрактной модели ассоциативных вычислений STAR-машины была доказана эквивалентность модели ASC. Основное направление для STAR-машины было в разработке ассоциативных алгоритмов, в частности для решения задач на графах.

Графические ускорители для решения задач на графах активно используются. Но в соответствующих системах и библиотеках представлены статические алгоритмы теории графов. Для STAR-машины разработаны также динамические алгоритмы, которые позволяют перестраивать решение после добавления и/или удаления ребер или вершин в графе. Параллельных динамических алгоритмов на графических ускорителях обнаружить не удалось.

Цель диссертационной работы состоит в построении эффективной реализации STAR-машины на графических ускорителях с использованием технологии CUDA. Это позволит использовать на практике ассоциативные параллельные алгоритмы, разработанные для этой модели.

Для достижения поставленной цели были решены следующие задачи:
- построена реализация базовых операций языка Star на графическом ускорителе;
- для увеличения эффективности реализации выделены операции языка Star, критичные к синхронизации;
- реализована на графическом ускорителе библиотека стандартных процедур языка Star;

[1] ВЦ СО АН СССР и далее в ИВМиМГ СО РАН, Непомнящая А.Ш.

[2] Kent State University, Department of Computer Science, Distributed and Parallel Processing

- на основании анализа существующих форматов входных/выходных данных для тестовых графов (более 5000 вершин) разработан модуль ввода/вывода данных отобранных форматов во внутреннее представление реализации Star-машины;
- обоснована эффективность реализации Star-машины как оценкой теоретической сложности процедур реализации, так и практическим сравнением времени работы с временем работы аналогов (для доказательства эффективности реализации базовых операций сравниваются время выполнения ассоциативной версии алгоритма Уоршалла с временем выполнения других параллельных реализаций этого алгоритма; для обоснования эффективности реализации библиотеки стандартных процедур проводится сравнение времени работы базовых процедур с временем работы аналогов из библиотек STL и CUDA thrust);
- разработаны методы оптимизации ассоциативных алгоритмов для выполнения на графических ускорителях, учитывающие различия Star-машины и GPU.

Научная новизна Предложена уникальная реализация модели ассоциативных вычислений на графических ускорителях: эффективно сохраняет ассоциативные свойства; расчитана на выполнение ассоциативных алгоритмов модели, а не прогнозирование ее свойств.

Для динамических алгоритмов решения задач теории графов нет неассоциативных параллельных алгоритмов, поскольку последовательные алгоритмы используют структуры данных, сложные для распараллеливания. Но использование данной технологии позволяет разрабатывать параллельные динамические алгоритмы.

Разработанные методы оптимизации ассоциативных алгоритмов для выполнения на графических ускорителях позволяют легко локализовать точки синхронизации в ассоциативных алгоритмах при реализации на GPU. Это значительно уменьшает трудозатраты разработчиков при их реализации.

Методология и методы исследования При получении основных результатов диссертационной работы использовались методы параллельного программирования, методы анализа информационной структуры параллельных алгоритмов, элементы теории графов, а также формальные модели оценки эффективности параллельных программ и степени локальности данных. При разработке реализаций графовых алгоритмов и создании программного комплекса использовались методы объектно–ориентированного анализа и проектирования, а также программно-аппаратная архитектура параллельных вычислений CUDA, а также средства анализа эффективности и производительности – nvprof.

Практическая значимость Для STAR-машины разработаны как классические, так и динамические алгоритмы для решения задач на графах. Реализация этих алгоритмов на графических ускорителях дает возможность их практического использования, сохраняя преимущества ассоциативной обработки.

Заметим также, что при распараллеливании алгоритмов возникают трудности с определением точек синхронизации. С одной стороны, неоптимальное расположение точек синхронизации потоков вычислений может сильно снизить производительность параллельной программы. С другой стороны, отсутствие необходимой точки синхронизации приводит к появлению ошибок, которые из-за большой степени недетерминизма параллельного исполнения трудно обнаружить традиционными методами отладки. Использование предложенной технологии значительно уменьшает трудозатраты разработчиков при разрабатывании и отладки параллельных алгоритмов.

На защиту выносятся следующие основные положения и результаты: Построена реализация абстрактной модели ассоциативной обработки данных (STAR-машины) на современной параллельной архитектуре (графических ускорителях).

Указаны операции языка Star, критичные к синхронизации.

Разработана классификация библиотеки стандартных процедур языка Star по способу обработки данных.

Выработаны методы оптимизации ассоциативных алгоритмов для выполнения на графических ускорителях, учитывающие различия Star-машины и GPU.

Эффективность реализации обоснована в теории и на примере выполнения ассоциативных алгоритмов.

Апробация работы Результаты работы обсуждались на семинарах «Математическое обеспечение высокопроизводительных вычислительных систем» ИВМиМГ СО РАН, «Дискретные экстремальные задачи» Института математики СО РАН, «Высокопроизводительные вычисления» ИВМиМГ СО РАН, лаборатории программных систем машинной графики ИАиЭ СО РАН, « ru-STEP по-русски» совместно Иннополис и ИСИ СО РАН, «Интеллектуальные системы и системное программирование» совместно ИСИ СО РАН и кафедры программирования НГУ. Основные результаты диссертации докладывались на конференциях МАРЧУКОВСКИЕ НАУЧНЫЕ ЧТЕНИЯ в 2017 и 2020 годах.

Публикации Материалы диссертации опубликованы в 8 печатных работах, из них в 4 статьях в научных журналах, входящих в перечень ВАК. Получено свидетельство о регистрации программ для ЭВМ.

Личный вклад автора Содержание диссертации и основные положения, выносимые на защиту, отражают персональный вклад автора в опубликованных работах. Подготовка к публикации полученных результатов проводилась совместно с соавтором. Соавтор претензий не имеет. Все представленные в диссертации результаты получены лично автором.

Структура и объем диссертации Диссертация состоит из введения, 3-х глав, заключения, списка сокращений и терминов, а также библиографии. Общий объем диссертации 114 страниц, из них 105 страницы текста, включая 28 рисунков. Библиография включает 79 наименований на 8 страницах.

Содержание работы

Во Введении обоснована актуальность диссертационной работы, сформулирована цель и аргументирована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные результаты.

В первой главе описана история развития ассоциативных архитектур от первого коммерчески успешного ассоциативного параллельного процессора STARAN до крупнейшего современного проекта с использованием ассоциативной архитектуры ATLAS Fast Tracker. Также приводятся описания моделей ассоциативных вычислений STAR-машины и MASC. История развития ассоциативных архитектур опубликована в работе [1].

Вторая глава посвящена реализации модели STAR-машины на графических ускорителях с помощью технологии CUDA. Выбор данной технологии для реализации STAR-машины обусловлен архитектурным сходством (рис. 1) и широким распространением устройств. К основным особенностям ассоци-

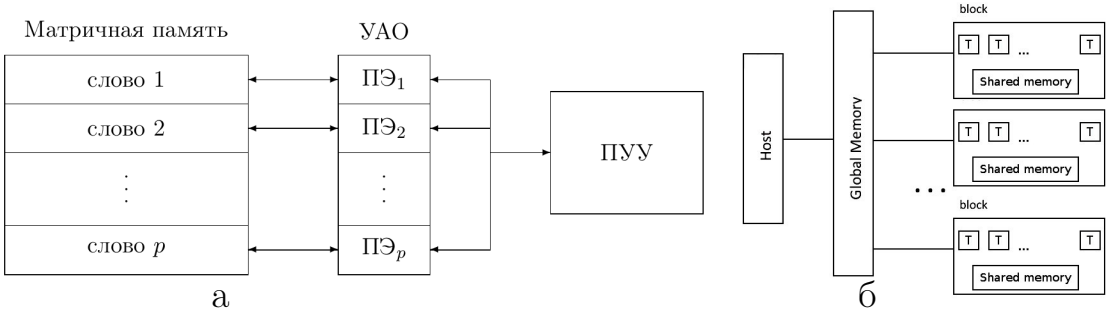

Рис. 1. Сравнение моделей: а) STAR-машина, б) GPU

ативных архитектур, отличающих их от архитектур фон-неймановского типа, относятся следующие: большое количество процессорных элементов (ПЭ); синхронное выполнение операций всеми активными ПЭ; передача данных от одного ПЭ всем другим за единицу времени.

Для реализации STAR-машины необходимо смоделировать типы данных и базовые операции языка Star, а также реализовать библиотеку базовых ассоциативных алгоритмов. При этом нужно убедиться, что ассоциативные свойства модели сохраняются.

Для доступа к матричной памяти в STAR-машине используются типы данных **word**, **slice** и **table**, для реализации которых используются одноименные классы. Заметим, что класс *Slice* используется и для типа **word**. Приведем описание этих классов.

Переменная типа Slice (битовый вектор) моделируется как массив элементов *unsigned long long int* в глобальной памяти GPU, при этом 64 элемента битового вектора хранятся в одном элементе массива. В классе Slice выделены поля length для хранения длины слайса в битах и NN для хранения длины массива.

Переменная типа Table моделируется как двумерный массив элементов *unsigned long long int* в глобальной памяти GPU с дополнительной системой указателей, позволяющей обращаться как к таблице целиком, так и к отдельному столбцу. Также выделены поля для хранения размеров: length (число строк) и size (число столбцов).

Базовые операции над переменными типа **slice** реализованы тремя способами: одноименный метод класса *Slice* для удобства использования; __global__ процедура, которая реализует операцию на GPU; __device__ аналог для базовых операций, не критичных к синхронизации, чтобы несколько таких операций можно было расположить в одном ядре. Таким образом, все вычисления производятся на графическом ускорителе параллельно и есть возможность минимизировать количество вызовов ядер.

Рассмотрим реализацию базовых операций над переменными типа **slice** более подробно. Для операций установки битов столбца $SET(X)$, $CLR(X)$, $MASK(X, i)$ и побитовых логических операций каждый из N потоков вычисляет свой элемент массива независимо от других. Поэтому время выполнения этих операций не зависит от длины битового столбца типа **slice**.

Рассмотрим реализацию операций выбора номера старшей ячейки $FND(X)$. На первом этапе массив из N элементов, представляющий слайс, сворачивается до одного числа по правилу, изображенному на рисунке 2. На втором

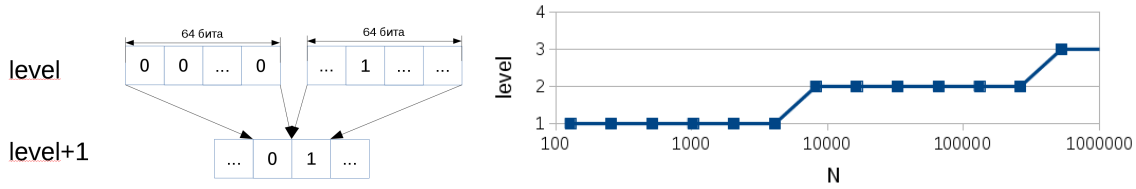

Рис. 2. Первый этап реализации операции FND и глубина свертки.

этапе определяется номер старшей ячейки. За счет применения такого приема время выполнения операций удалось свести к $O(\lceil \log_{64}(N) \rceil)$, близкое к константному.

Оператор $STEP(X)$ также возвращает номер старшей единицы в слайсе X, после чего меняет значение в этой позиции на 0. Этот оператор, а также предикаты для слайсов $SOME(X)$ и $ZERO(X)$ выполняются через реализацию оператора $FND(X)$.

Операции над переменными типа **table** $COL(i, T)$ и $ROW(i, T)$ выполняются за константное время. Тем не менее $ROW(i, T)$ выполняется медленнее и является критичной к синхронизации.

Таким образом, базовые операции языка Star представлены в виде процедур, выполняемых на графическом ускорителе за константное время или за время $O(\lceil \log_{64}(N) \rceil)$, близкое к константному. Критичными к синхронизации являются следующие операции: $FND(X)$, $STEP(X)$, $SOME(X)$, $ZERO(X)$, $NUMB(X)$ и $ROW(i, T)$.

Операция	Процедура	Время выполнения (μs)			
		100	1 000	3 000	5 000
ROW (чтение)	get_row	18,6 (18,5 – 19,7)	19, 5	19,6	19,8
ROW (запись)	set_row	4,1 (2,9 – 5,6)	3,9	4,3	4,1
or (побитовое)	or_long_value	1,9 (1,9 – 3,0)	2,2	2,0	2,1
FND (номер старшей единицы)	find	2,5 (2,3 – 2,8)	2,8	3,5	3,8
	first_backward	3,2 (3,1 – 4,0)	3,2	3,3	3,4

Таблица 1. Профилирование базовых операций на узле k40 (Kepler) кластера nks-30T ССКЦ.

Результаты профилирования показаны в таблице 1. В первом столбце приведено название базовой операции, во втором — имя процедуры, выполняющейся на графическом ускорителе (реализации операции FND вызывает два ядра). Здесь время выполнения процедуры включает в себя время за-

пуска ядра, поэтому может существенно отличаться для значений порядка нескольких микросекунд. Отметим также, что операция COL передает указатель на столбец, поэтому ее время работы не измеряется. В таблице 1 приведено среднее время запуска, а также для 100 вершин в скобках показаны минимальное и максимальное время выполнения процедур. Таким образом, время выполнения базовых операций практически не зависит от размера данных, что соответствует теоретическим оценкам: $O(\log_{64}(N))$ для реализации операции $FND(X)$, $STEP(X)$, $SOME(X)$ и $ZERO(X)$, и константное время для реализации остальных операций.

Ниже покажем результаты реализации библиотеки стандартных ассоциативных алгоритмов. При реализации библиотеки алгоритмы естественным образом разделяются на следующие группы по способу работы с таблицами:
I используются базовые операции, критичные к синхронизации: MIN и MAX;
II не используются базовые операции, критичные к синхронизации, при этом есть зависимость данных по столбцам: поиск строк таблицы, совпадающих с образцом MATCH, сравнение строк таблицы с образцом GREAT, LESS, GEL, сравнение двух таблиц построчно SETMIN, SETMAX, HIT;
III арифметические алгоритмы: прибавление слова к строкам таблицы ADDC, построчное сложение таблиц ADDV, вычитание слова из всех строк таблицы SUBTC, построчное вычитание одной таблицы из другой SUBTV;
IV не используются базовые операции, критичные к синхронизации, при этом нет зависимости данных по столбцам: CLEAR, TMERGE, WMERGE, WCOPY, TCOPY, TCOPY1, TCOPY2.

Группа	Оценка сложности		Размер данных	Время μs	
алгоритмов	алгоритма	реализации	(одновременно)	GF920m	k40
I	$O(h)$	$O(h \cdot \lceil \log_{64}(N) \rceil)$	до 4 096 4 097 – 100 000	400 570 – 590	713 865 – 890
II	$O(h)$	$O(h)$	100 000	58 – 61	71 – 73
III	$O(h)$	$O(h + \lceil \log_{64}(N) \rceil)$	50 000	36 – 41	52 – 58
IV	$O(h)$	$O(1)$	10 000	6-10	8-9

Таблица 2. Теоретическая сложность и время выполнения реализации библиотеки стандартных алгоритмов.

В таблице 2 приведены теоретическая оценка сложности базовых алго-

ритмов и их реализаций, а также время выполнения реализаций на графических ускорителях . Здесь h - ширина таблицы, N - длина таблицы. Расчеты проводились на карте NVIDIA GEFORCE 920M и узле k40 (Kepler) кластера nks-30T ССКЦ.

Сравнение времени работы реализации библиотеки стандартных ассоциативных параллельных алгоритмов с временем работы подобных алгоритмов из библиотек C++ STL (CPU) и CUDA thrust (GPU) показало, что созданная реализация базовых ассоциативных алгоритмов дает ускорение до 2 порядков на векторах от 5000 элементов по сравнению с библиотекой STL и в 1,5-2 раза по сравнению с библиотекой CUDA thrust.

Во второй главе также даются рекомендации по оптимизации ассоциативных алгоритмов для выполнения на графических ускорителях. Они следуют из отличий ассоциативных архитектур от графических ускорителей и состоят из трех шагов. На первом шаге необходимо определить точки синхронизации и проверить, можно ли уменьшить их число. На втором шаге нужно определить, есть ли зависимость по данным при обработке столбцов. На последнем шаге рекомендуется уменьшить количество вызовов ядер. Для этого операторы и базовые процедуры II-IV групп, расположенные между точками синхронизации, объединяют в минимальное число ядер.

Результаты второй главы были опубликованы в работах [2, 3],[7].

В **третьей главе** приводятся три ассоциативных алгоритма, демонстрируется их оптимизация и результаты вычислительных экспериментов.

Первым из алгоритмов рассмотрим ассоциативную версию **алгоритма Уоршалла** поиска транзитивного замыкания. Данный алгоритм по матрице смежности графа строит матрицу достижимости графа между всеми парами вершин. При оптимизации ассоциативного алгоритма Уоршалла для вычисления на GPU были построены две версии алгоритма. Ассоциативная версия WARSHALL-C производит обработку по столбцам с теоретической оценкой сложности $O(n^2)$. Адаптированная версия WARSHALL-adapt обра-

батывает столбцы параллельно, поскольку нет зависимости по данным между столбцами. Теоретическая оценка сложности для этой версии уменьшается до $O(n)$. Как видно из результатов вычислительного эксперимента (таблица 3)

Количество вершин (n)	1 000	2 000	3 000	4 000	5 000
Последовательный алгоритм	8,037	59,812	197,111	461,785	884,622
WARSHALL-C	4,827	11,161	23,363	42,661	64,454
WARSHALL-adapt	**0,003**	**0,027**	**0,093**	**0,236**	**0,372**

Таблица 3. Время работы последовательного алгоритма Уоршалла и его Star-версий на графической карте GeForce920m.

использование адаптированного ассоциативного алгоритма Уоршалла позволяет уменьшить время расчета на несколько порядков. Сравнительное время расчета ассоциативных версий алгоритма Уоршалла для графов, имеющих 5000 вершин, на различных графических ускорителях показано на рисунке 3.

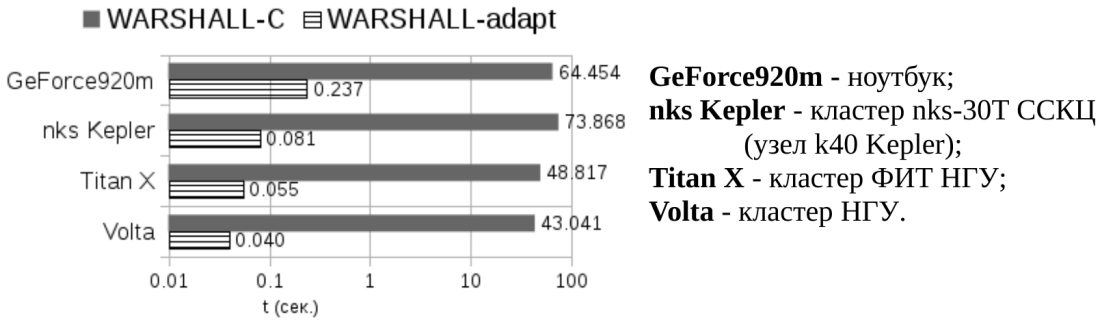

Рис. 3. Время работы ассоциативных версий для графа, имеющего 5000 вершин.

Теперь рассмотрим работу двух динамических алгоритмов. Такие алгоритмы позволяют после локального изменения в графе получать решение, учитывающее имеющееся решение для предыдущего состояния графа. Таким образом, динамические алгоритмы выполняются быстрее, чем поиск решения для измененного графа самым быстрым статическим алгоритмом.

Первым из динамических алгоритмов рассмотрим **ассоциативный алгоритм для динамической обработки дерева кратчайших путей после добавления дуги**. Данный алгоритм приведен в работе [4]. Оценка производительности проводилась на R-MAT-графах. Они хорошо моделируют реальные графы из социальных сетей и интернета, а также являются достаточно сложными для анализа.

В экспериментах использовались два режима: **R-MAT*** — добавлялись дуги с весом 0 (пессимистичный сценарий); **R-MAT** — добавлялись дуги со случайным весом (реалистичный сценарий). В большинстве случаев (более 50 % при добавлении дуги с нулевым весом и более 88 % — со случайным весом) кратчайшие пути и расстояния не изменяются. В 99 % случаев число аффектных вершин (до которых кратчайшее расстояние уменьшается) не превышает 5 в первом режиме и 2 во втором. Отметим, что в отличие от ассоциативного алгоритма, ведущего обход графа по вершинам (исходящие дуги обрабатываются параллельно), в последовательном варианте обход графа совершается по дугам, и число дуг, которые необходимо проверить, может доходить до 2000 в первом режиме и до 100 во втором.

В таблице 4 приводятся среднее время выполнения процедуры DistSPT (статический ассоциативный алгоритм) и максимальное время выполнения процедуры InsertArcSPT (динамический ассоциативный алгоритм) в двух режимах добавления дуг. Здесь n — число вершин в графе, k — число аффектных вершин в худшем случае (число аффектных вершин при добавлении дуги с максимальным временем обработки/максимальное число аффектных вершин). Время и количество аффектных вершин для режима R-MAT* отмечены «*».

Граф	n	DistSPT	InsertArcSPT*	k*	InsertArcSPT	k
R-MAT-11	2048	6,171	0,012	6/8	0,009	1/8
R-MAT-12	4096	9,643	0,017	5/10	0,012	3/10
R-MAT-13	8192	29,475	0,038	7/15	0,020	4/13

Таблица 4. Сравнение времени работы статического и динамического ассоциативных алгоритмов на R-MAT-графах

Таким образом, при использовании динамического алгоритма время определения кратчайших расстояний уменьшается на несколько порядков, так как InsertArcSPT зависит от числа аффектных вершин, но не зависит от общего числа вершин в графе.

В заключение рассмотрим **ассоциативную версию динамического алгоритма Рамалингама для проблемы достижимости в потоковом**

графе с одним источником. Этот алгоритм подробно описан в работе [5].

В ходе тестирования время работы динамического ассоциативного алгоритма (D_A) сравнилось со временем работы статического ассоциативного алгоритма (S_A). А также подсчитывались изменение числа достижимых вершин после добавления дуги и число дуг, исходящих из вершин, которые стали достижимыми после добавления дуги. Отметим, что эти величины определяют число итераций ассоциативного (D_A) и последовательного (D_S) динамических алгоритмов, соответственно.

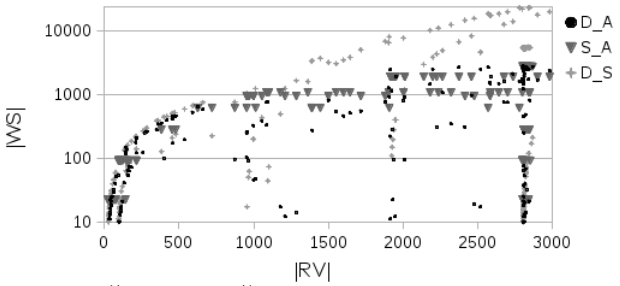

Рис. 4. Количество итераций (|WorkSet|), необходимых для обработки графа после добавления дуги, заходящей в вершину v.

На рисунке 4 приводится количество итераций, которое требуется для обработки заданного графа после добавления новой дуги. На рисунке видно, что число итераций для динамического ассоциативного алгоритма и динамического последовательного алгоритма могут значительно различаться в зависимости от структуры обрабатываемого графа. Полученное ускорение зависит от усредненной степени вершин графа и исходного множества достижимых вершин. Отметим, что динамический алгоритм выполняется до 10 раз быстрее, чем статический. Результаты третьей главы опубликованы в работах [2, 4] и [5],[8].

Заключение Таким образом можно заключить, что реализация Star-машины на графических ускорителях является актуальной, эффективной и дает возможность использовать ассоциативные алгоритмы на практике. При этом в отличии от распараллеливании алгоритмов, не возникает трудности с определением точек синхронизации. Это связано с тем, что в языке Star выявлены операции, критичные к синхронизации, и выработаны методы оп-

тимизации алгоритмов для выполнение на GPU.

В качестве перспективы дальней разработки тематики рассматривается приложение Star-машины и ее реализации на GPU к задачам биоинформатики. Эти задачи отличаются чертами для которых ассоциативная обработка является оптимальной: поиск по большому массиву данных: от десятков тысяч до нескольких миллиардов нуклеотидных оснований; ограниченный алфавит: A,T,C,G для нуклеотидов (3 бита) и 20 для аминокислот (5 битов); управляющие слайсы дают возможность троичного поиска.

Публикации автора по теме диссертации

1. Снытникова Т. В. Развитие ассоциативных параллельных архитектур // Проблемы информатики. 2019. № 2. С. 36–50.
2. Снытникова Т. В., Непомнящая А. Ш. Решение задач на графах с помощью STAR-машины, реализуемой на графических ускорителях // Прикладная дискретная математика. 2016. Vol. 3(33). P. 98–115.
3. Снытникова Т. В. Реализация модели ассоциативных вычислений на GPU: библиотека базовых процедур языка STAR // Вычислительные методы и программирование. Новые вычислительные технологии. 2018. Vol. 19. P. 85–95.
4. Непомнящая А. Ш., Снытникова Т. В. Ассоциативный параллельный алгоритм для динамической обработки дерева кратчайших путей после добавления новой дуги // Прикладная дискретная математика. 2019. Vol. 46. P. 49–62.
5. Непомнящая А. Ш., Снытникова Т. В. Ассоциативная версия инкрементального алгоритма Рамалингама для решения проблемы достижимости в потоковых графах с одним источником // Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика. 2021. Vol. 54. P. 86–96.
6. Свидетельство о государственной регистрации программы ЭВМ, базы данных, зарегистрированных в Роспатенте № 2021665653 Российская Федерация. cuSTAR / Т. В. Снытникова; заявитель и патентообладатель ИВМиМГ СО РАН. - № 2021665653; заявл. 14.09.21; опубл. 20.10.21. Бюль. № 10. - 1 С.
7. Снытникова Т. В., Непомнящая А. Ш. О реализации на GPU базовых ассоциативных процедур языка STAR // Марчуковские научные чтения. Vol. 2017. 2017.
8. Снытникова Т. В., Непомнящая А. Ш. Реализация на GPU инкрементального алгоритма Рамалингама для динамической обработки потоковых графов с одним источником // Марчуковские научные чтения. Vol. 2020. 2020.