



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ

КАФЕДРА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И СИСТЕМНОГО ПРОГРАММИРОВАНИЯ

ЛАБОРАТОРИЯ ITLav



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ
КАФЕДРА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И СИСТЕМНОГО ПРОГРАММИРОВАНИЯ
ЛАБОРАТОРИЯ ITLAB

***ВВЕДЕНИЕ В АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ
И ОПТИМИЗАЦИЮ ПРОГРАММ***

**Параллельные алгоритмы
раскраски графов**



**LOBACHEVSKY
UNIVERSITY**

Воденеева А.А., Мееров И.Б., Пирова А.Ю.

Содержание

- ❑ Введение
- ❑ Методы раскраски вершин графа
- ❑ Параллельные алгоритмы
- ❑ Программная реализация
- ❑ Результаты экспериментов
- ❑ Заключение

ВВЕДЕНИЕ

Введение

- Пусть дан граф:

$$G = (X, U),$$

X – множество вершин данного графа, U – множество его ребер.

- Требуется найти разбиение множества вершин X на L непересекающихся подмножеств:

$$X_1, X_2, \dots, X_L;$$

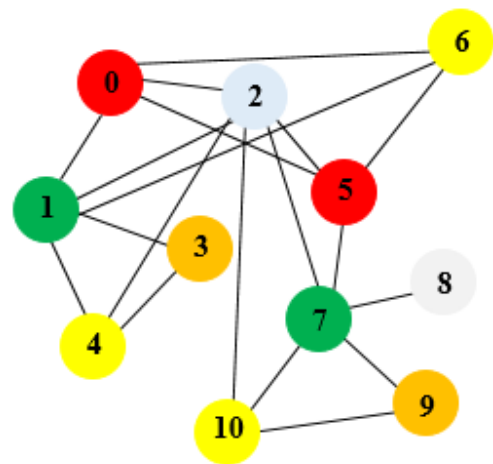
$$X = \bigcup_{i=1}^L X_i;$$

$$X_i \cap X_j = \emptyset, i, j = 1, 2, \dots, L,$$

чтобы внутри каждого подмножества X_i не содержалось смежных вершин.

Введение

- Каждому из этих подмножеств ставится в соответствие определенный цвет: вершины внутри одного подмножества можно раскрасить в один цвет, вершины внутри другого подмножества – в другой цвет, и так далее до раскраски всех L подмножеств



Порядок раскраски вершин: 2, 1, 7, 0, 5, 4, 6, 10, 3, 9, 8

Порядок цветов: 1 – голубой, 2 – зеленый, 3 – красный, 4 – желтый, 5 – оранжевый, 6 – серый

Введение

- ❑ Цель работы – сравнительный анализ подходов к распараллеливанию вычислений при решении задачи о раскраске графа с использованием современных центральных и графических процессоров Intel
- ❑ Для этого:
 - Изучить параллельные алгоритмы раскраски графа и выполнить программную реализацию наиболее перспективных алгоритмов с использованием языков и библиотек C++/OpenMP, Data Parallel C++, KOKKOS
 - Сравнить производительность разработанных кодов программ на доступных CPU и GPU и выявить основные архитектурные механизмы, которые влияют на производительность, улучшить код в соответствии с результатами анализа

МЕТОДЫ РАСКРАСКИ ВЕРШИН ГРАФА

Методы раскраски вершин графа

Алгоритмы раскраски графов



```
graph TD; A[Алгоритмы раскраски графов] --> B[Последовательные алгоритмы]; A --> C[Параллельные алгоритмы]; B --> D[жадный алгоритм]; B --> E[стягивание]; B --> F[полиномиальный алгоритм]; C --> G[алгоритм Джонса-Плассмана]; C --> H[алгоритм Чаталюрека]; C --> I[алгоритм Боумана];
```

Последовательные алгоритмы

жадный алгоритм

стягивание

полиномиальный
алгоритм

Параллельные алгоритмы

алгоритм
Джонса-Плассмана

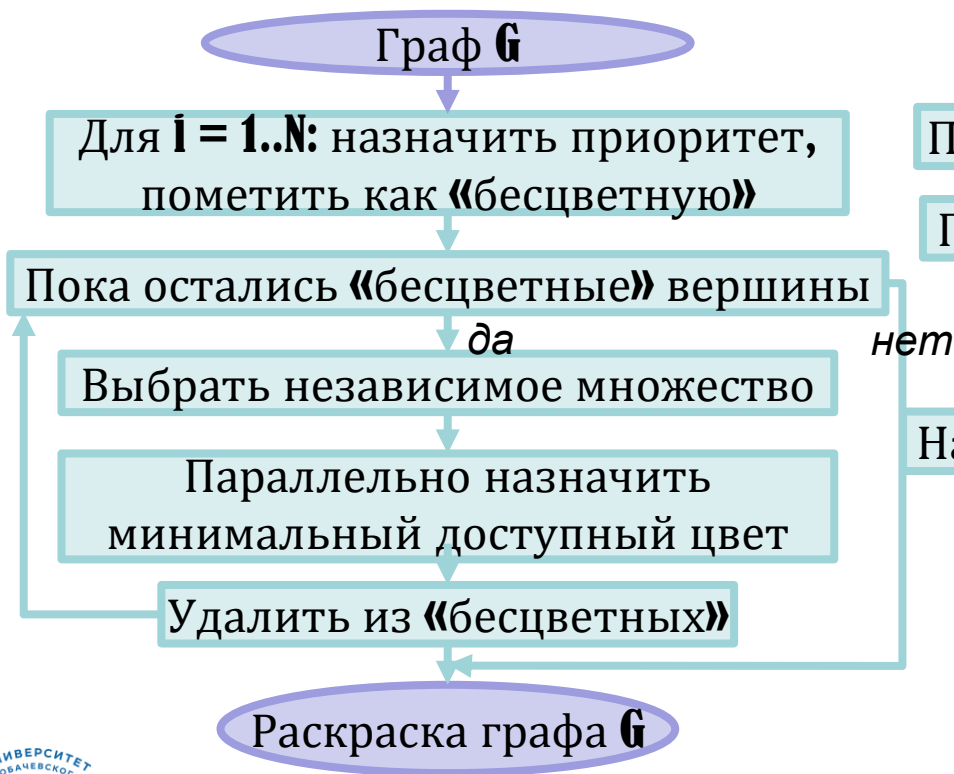
алгоритм Чаталюрека

алгоритм Боумана

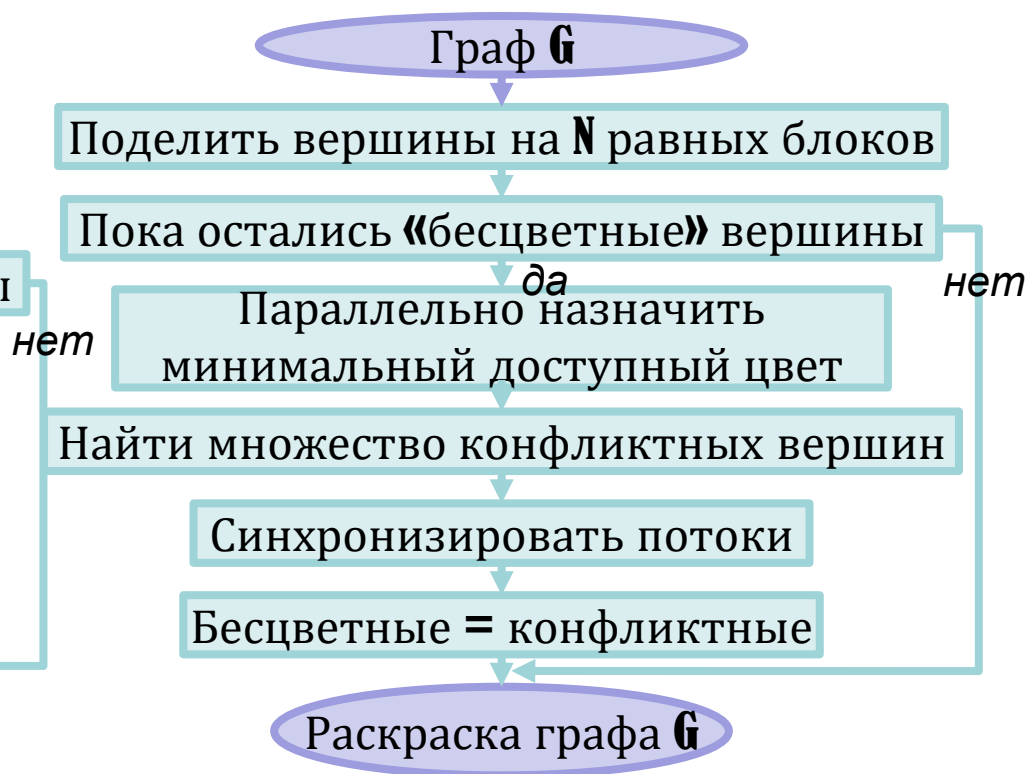
ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ

Параллельные алгоритмы

□ Алгоритм Джонса-Плассмана



□ Алгоритм Чаталюрека



ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Программная реализация

- ❑ Выполнена на языке C++, в среде MS Visual Studio 2019
- ❑ Общее число строк кода: 1614
- ❑ Параллельные версии алгоритмов Джонса-Плассмана и Чаталюрека реализованы с использованием технологий OpenMP, KOKKOS и Data Parallel C++.
- ❑ OpenMP – открытый стандарт для распараллеливания программ на языках C, C++ и Fortran.
- ❑ KOKKOS – модель программирования на C++ для написания высокопроизводительных переносимых приложений, ориентированных на все основные платформы.
- ❑ Data Parallel C++ – модель параллельного программирования с использованием новейших стандартов C++.

Программная реализация

- ❑ Для алгоритма Чаталюрека во всех трех реализациях была произведена оптимизация производительности для GPU за счет изменения структуры данных:
 - массив конфликтных цветов, получаемых для шага разрешения конфликтов, был заменен на битовые сдвиги: каждое добавление конфликтного цвета в массив было заменено на битовый сдвиг по номеру данного цвета.
- ❑ Для алгоритма Чаталюрека в реализациях с использованием моделей KOKKOS и Data Parallel C++ было применено кэширование данных в объемных по рабочему пространству циклах.

МЕТОДИКА ПРОВЕДЕНИЯ ЭКСПЕРИМЕНТОВ

Методика проведения экспериментов

- ❑ Тестовые матрицы (*коллекция SuiteSparse*)^[1]:

Номер	Название	Количество вершин	Номер	Название	Количество вершин
1	one34by	259 789	6	pre36	5 012 457
2	bcsstk12	715 176	7	raw34	6 456 202
3	bcsstk3	952 203	8	dersame	8 222 012
4	compronb	1 391 349	9	mianse2	11 054 532
5	will12	1 564 794	10	nu16ddk	14 758 344

- ❑ Тестовые системы:

- узел кластера Intel DevCloud с центральным процессором Intel Xeon;
- узел кластера Intel DevCloud с дискретным графическим процессором Intel Iris XE Max.

[1] <https://sparse.tamu.edu/>

Методика проведения экспериментов

- ❑ Для анализа эффективности проведено сравнение со встроенными функциями библиотек *ColPack*^[1] и *KOKKOS*^[2]
- ❑ Проведено сравнение эффективности и качества раскраски для полученных алгоритмов друг с другом в различных реализациях (OpenMP, KOKKOS, Data Parallel C++)
- ❑ Для алгоритма Чаталюрека проведено сравнение эффективности реализаций в запусках на центральном и графическом процессорах

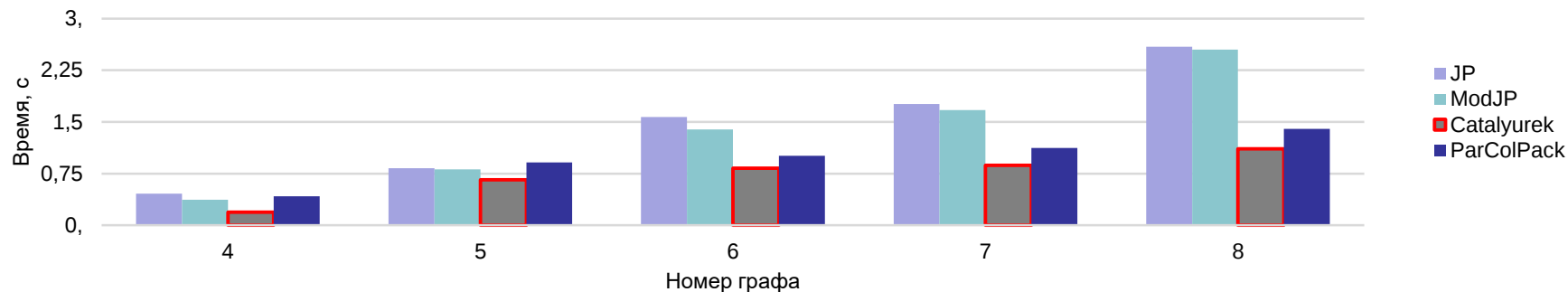
[1] <https://cscapes.cs.purdue.edu/coloringpage/>

[2] <https://github.com/kokkos>

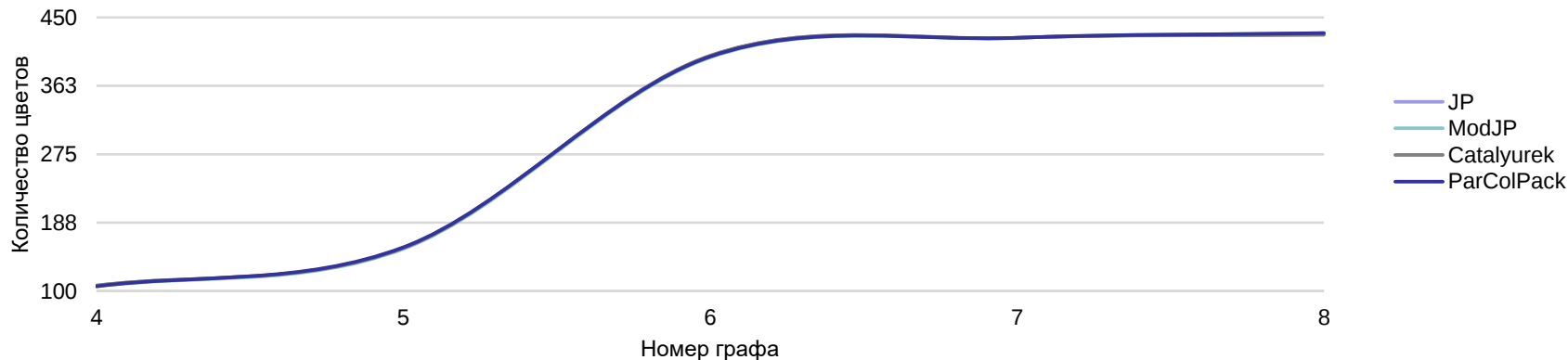
РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Результаты экспериментов

Время работы реализаций на OpenMP

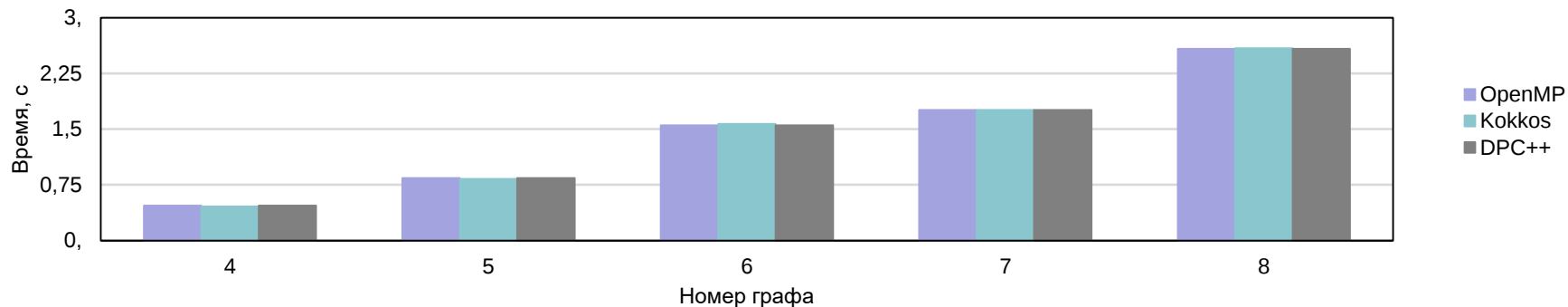


Количество цветов раскраски

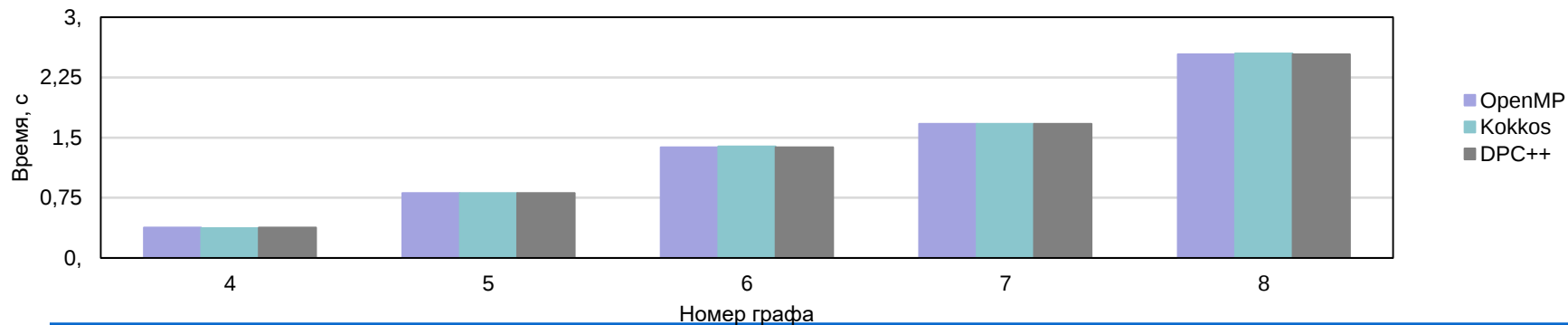


Результаты экспериментов

Время работы реализаций алгоритма Джонса-Плассмана

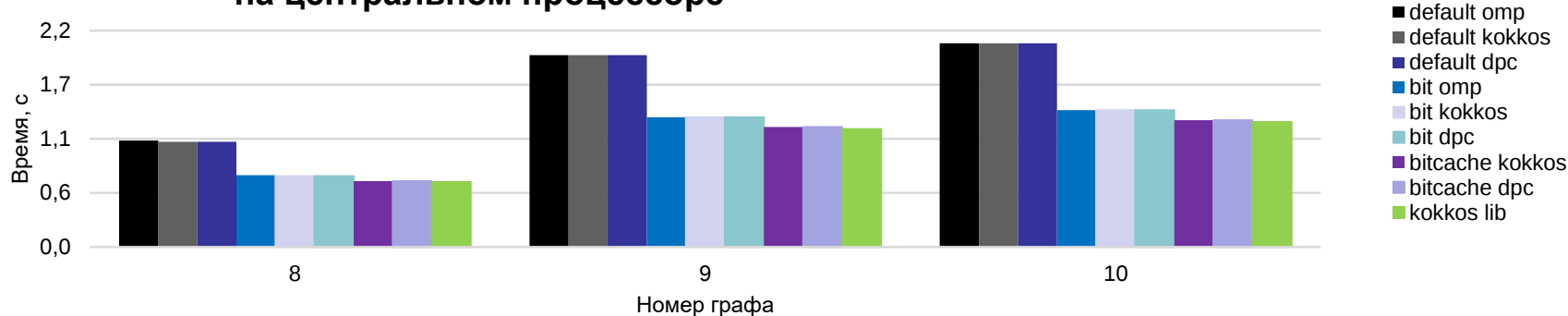


Время работы реализаций модифицированного алгоритма Джонса-Плассмана

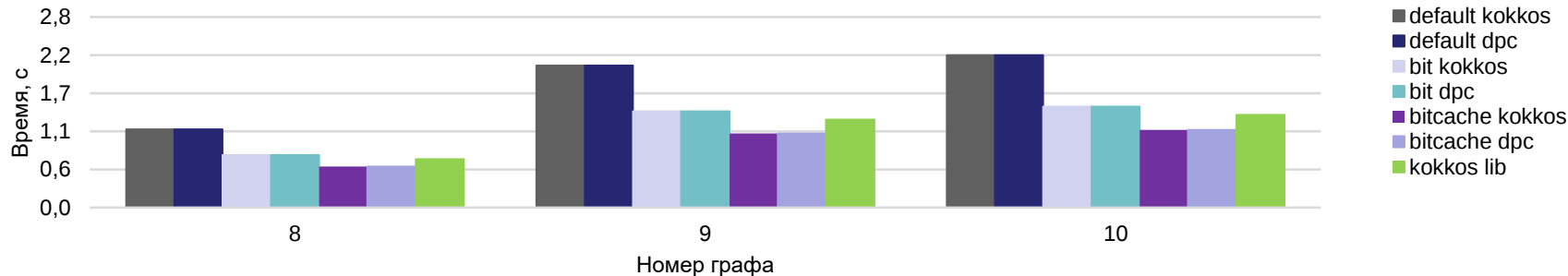


Результаты экспериментов

Время работы всех реализаций алгоритма Чаталюрека на центральном процессоре



Время работы реализаций на KOKKOS и Data Parallel C++ алгоритма Чаталюрека на графическом процессоре



ЗАКЛЮЧЕНИЕ

Заключение

- ❑ Алгоритмы раскраски были реализованы с использованием трех различных моделей программирования – OpenMP, KOKKOS и Data Parallel C++, две последних позволили создать переносимые параллельные реализации для расчетов на центральном и графическом процессорах
- ❑ Все три модели программирования позволили создать очень близкие по производительности версии параллельных алгоритмов раскраски
- ❑ Битовые операции и кэширование позволили оптимизировать исходную реализацию алгоритма Чаталюрека и получить ускорение для графического процессора, на центральном процессоре результаты работы модифицированной версии также улучшились

Контакты

Нижегородский государственный университет

<http://www.unn.ru>

Институт информационных технологий, математики и механики

<http://www.itmm.unn.ru>

Кафедра высокопроизводительных вычислений и системного программирования

