УНИВЕРСИТЕТ ЛОБАЧЕВСКОГО

Реализация алгоритмов раскраски графов с использованием SYCL и KOKKOS

А.А. Курникова, А.Ю. Пирова, В.Д. Волокитин, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского

Цели и задачи

УНИВЕРСИТЕТ ЛОБАЧЕВСКОГО
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Цель работы: разработать переносимую параллельную реализацию алгоритма Чаталюрека для раскраски графа и изучить, как различные технологии распараллеливания и приемы оптимизации кода влияют на производительность на центральных и графических процессорах.

Для этого:

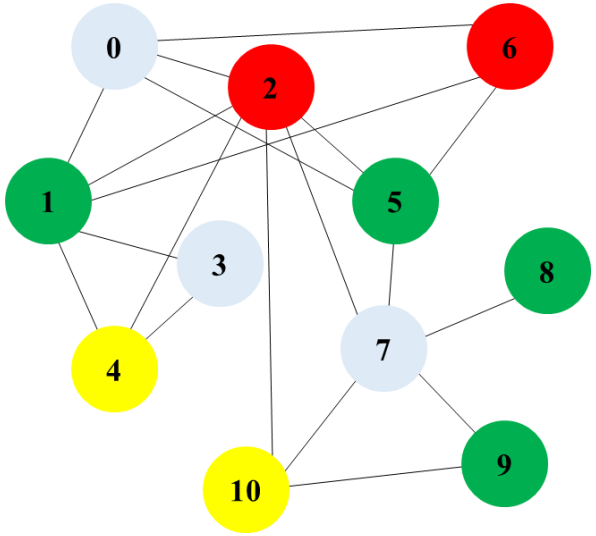
- параллельный алгоритм Чаталюрека раскраски графа был изучен и реализован с использованием OpenMP/C++, KOKKOS и SYCL;
- произведен сравнительный анализ производительности разработанных кодов программ на CPU и GPU;
- выявлены основные архитектурные механизмы, влияющие на производительность;
- произведено улучшение кода в соответствии с результатами анализа.

Задача раскраски

УНИВЕРСИТЕТ ЛОБАЧЕВСКОГО
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

- Граф $G = (X, U)$, X – множество вершин данного графа, U – множество его ребер.
- Находится разбиение множества вершин X на такие непересекающиеся подмножества, внутри каждого из которых не содержится смежных вершин.
- Каждому подмножеству ставится в соответствие определенный цвет.

.



Алгоритм Чаталюрека:

- Подходит для любой системы с общей памятью, в том числе для современных многоядерных платформ.

Инструменты

УНИВЕРСИТЕТ ЛОБАЧЕВСКОГО
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

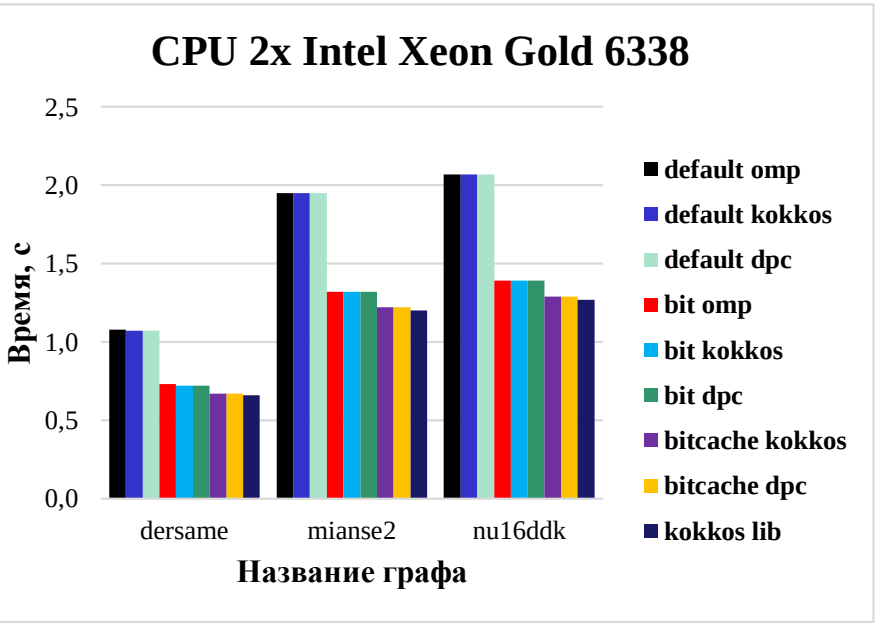
- **KOKKOS C++ Performance Portability EcoSystem:**
  - модель программирования на C++ для написания высокопроизводительных переносимых приложений, ориентированных на все основные платформы;
  - включает модель программирования, математические библиотеки и инструменты профилирования и отладки.

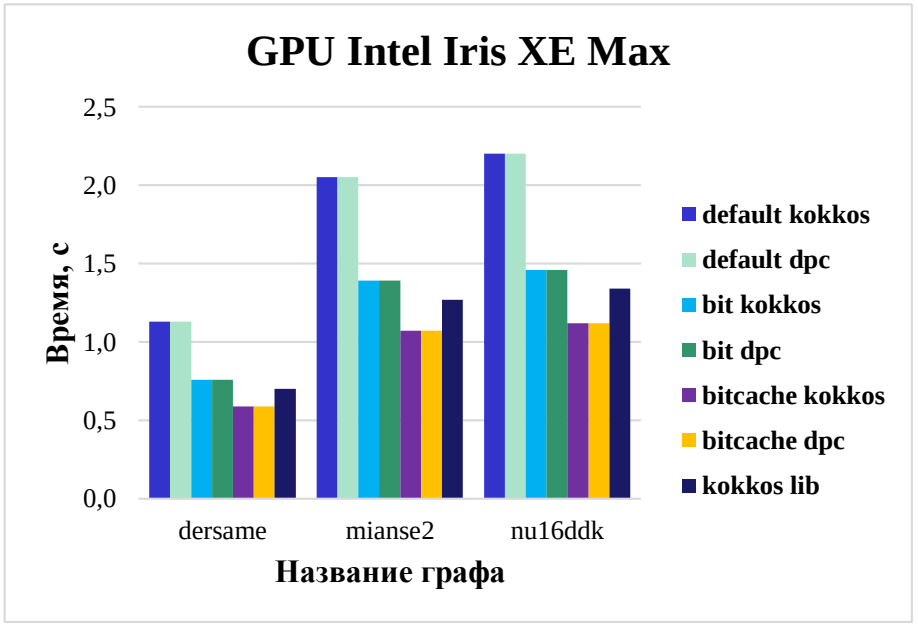
- **SYCL (реализация Intel OneAPI – Data Parallel C++):**
  - модель параллельного программирования, основанная на новейших стандартах языка C++;
  - включает несколько расширений и широкую гетерогенную поддержку для устройств GPU, CPU и FPGA.

- 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.

- В реализациях с использованием моделей KOKKOS и SYCL было применено кэширование данных в объемных по рабочему пространству циклах.

Результаты





- При решении задачи раскраски графа модели программирования KOKKOS и Data Parallel C++ (SYCL) показали схожие результаты и не уступили комбинации C++/OpenMP в случаях, когда такое сравнение возможно.
- При решении данной задачи они позволили использовать код для расчетов как на центральном, так и на графическом процессоре без существенной потери производительности.