Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра компьютерных технологий

Применение графических процессоров для генерации управляющих атвоматов на основе моделирования и сценариев работы с помощью эволюционных алгоритмов

Автор доклада: Бочкарев А.И.

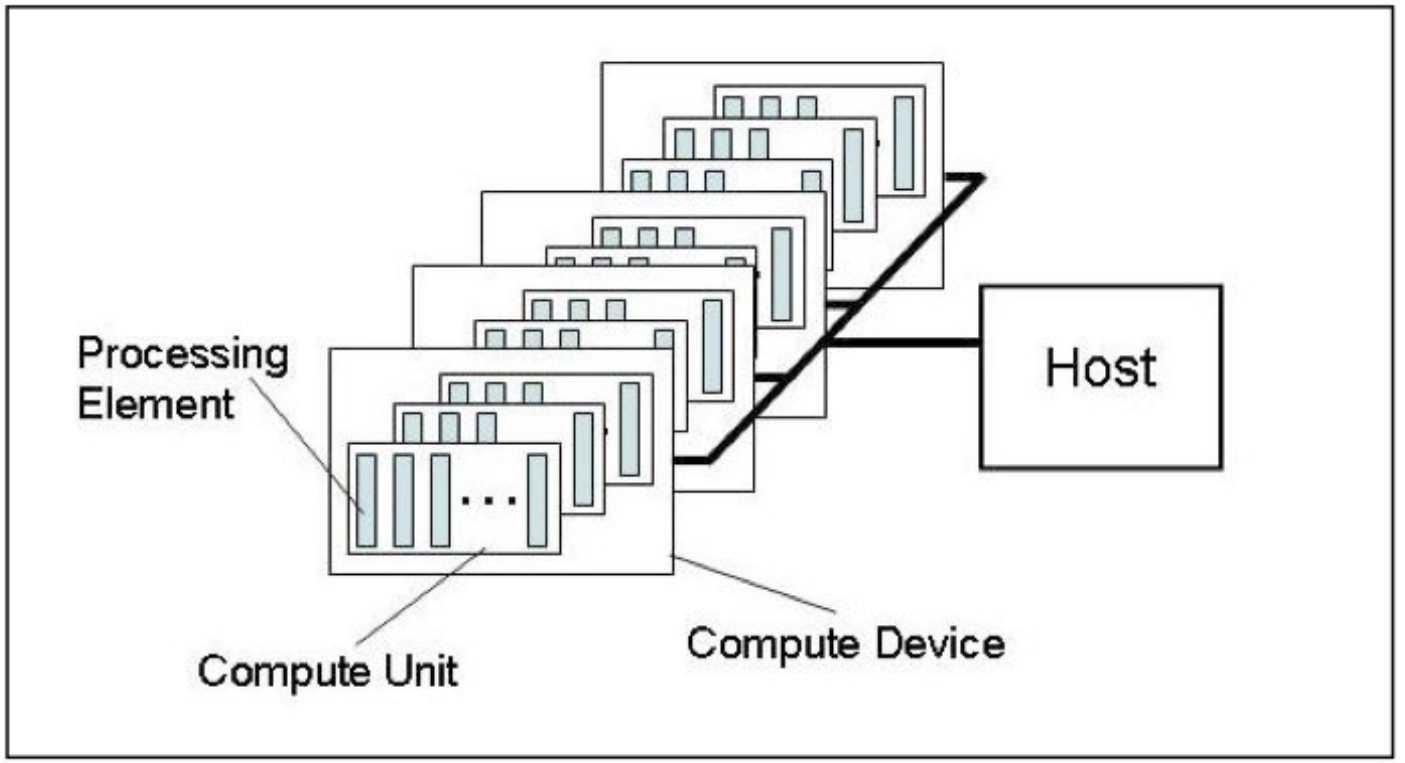
Научный руководитель: Шалыто А.А.

Цель работы

- Применение графических процессоров при генетическом построении управляющих автоматов в различных задачах
- Изучение платформы *OpenCL* и экспериментальное сравнение ее производительности со стандартными технологиями многопоточности
- Исследование вычислительных способностей графических процессоров
- Изучение применимости полученных результатов в других задачах

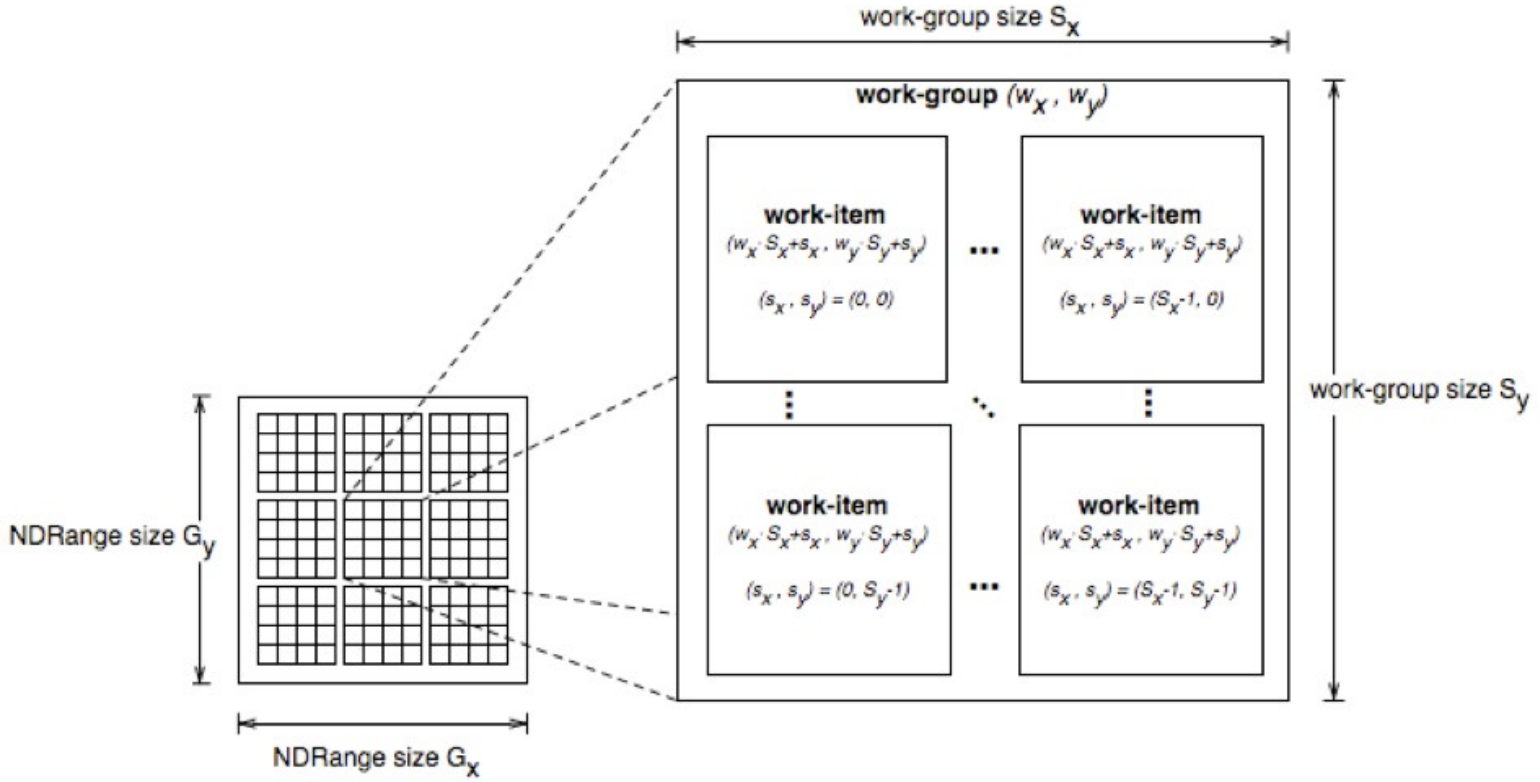
Модель платформы *OpenCL*

- Каждая реализация *OpenCL* определяет платформы, которые позволяют приложению взаимодействовать с вычислительными устройствами
- *OpenCL* использует технологию «Installable Client Driver»
- Структура платформы *OpenCL* изображена на рисунке:



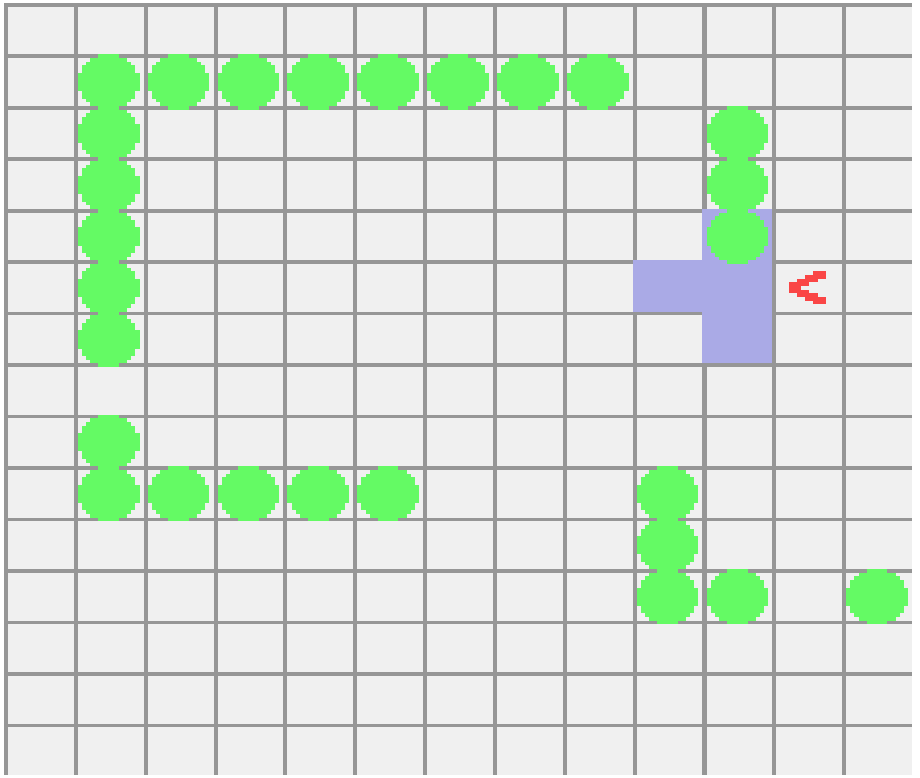
Структура потока выполнения

- Каждый экземпляр ядра называют рабочей единицей
- Рабочие единицы объединяют в рабочие группы
- Число рабочих единиц в группе ограничено и зависит от конкретного устройства
- Все пространство вычислений представляет из себя объединение рабочих групп



Серия задач об умном муравье

- Поле игры представляет двумерный тор NxM
- В некоторых клетках расположены яблоки
- Муравей имеет область видимости, которая варьируется в различных модификациях
- За один ход муравей может повернуться или сделать шаг вперед, при этом он съедает яблоко, если оно находилось прямо перед ним
- Необходимо построить управляющий автомат, который максимизирует число съеденных яблок за некоторое число ходов

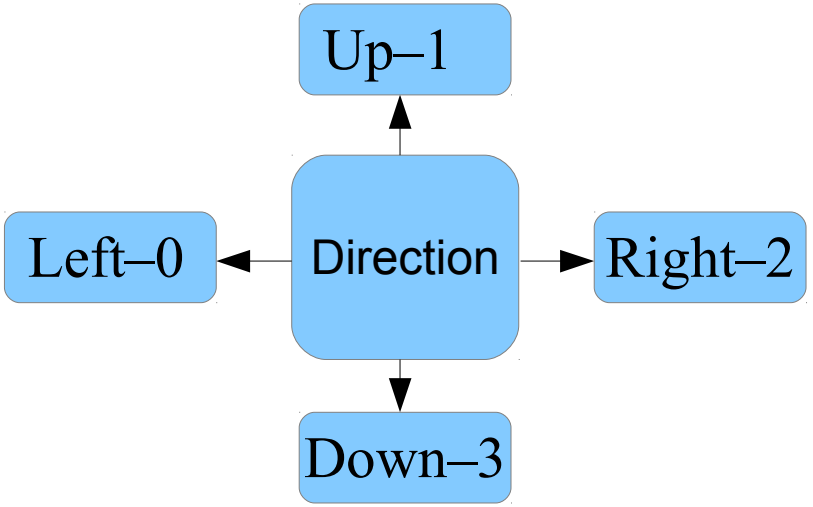


Построение автоматов в данной работе

- Входные воздействия – есть ли яблоко в той или иной клетке из области видимости
- Автомат хранится в виде полной таблицы переходов, сокращенной таблицы и дерева решений
- Используются различные генетические алгоритмы
- Размеры поколений варьируются в экспериментах
- Используется стандартная функция приспособленности: число съеденных муравьем яблок за N ходов

Особенности реализации

- Условные переходы заменены на арифметику и присвоение при условии




```
uint left( uint direction )
{
    return (direction-1)&3;
}
uint right( uint direction )
{
    return (1+direction)&3;
}
```

```
uint actionTurn( uint direction, uint action )
{
    int res = select( 0, -1, action == 2 );
    res = select( res, 1, action == 1 );
    return (direction+res)&3;
}
```

Случайные числа

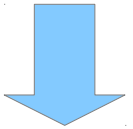
- Линейный генератор псевдослучайных чисел
- Обычная и векторная реализации

```
__constant uint rand_a = 214013;
__constant uint rand_c = 2531011;

uint nextInt( uint x )
{
    return ( rand_a*x + rand_c );
}

char16 nextUChar16( char16 x )
{
    char16 a = (char16)(17);
    char16 b = (char16)(31);
    return mad_hi( x, a, b );
}
```

Скрещивание особей

- В операции скрещивания, как и в мутации, скалярные операции заменены на векторные

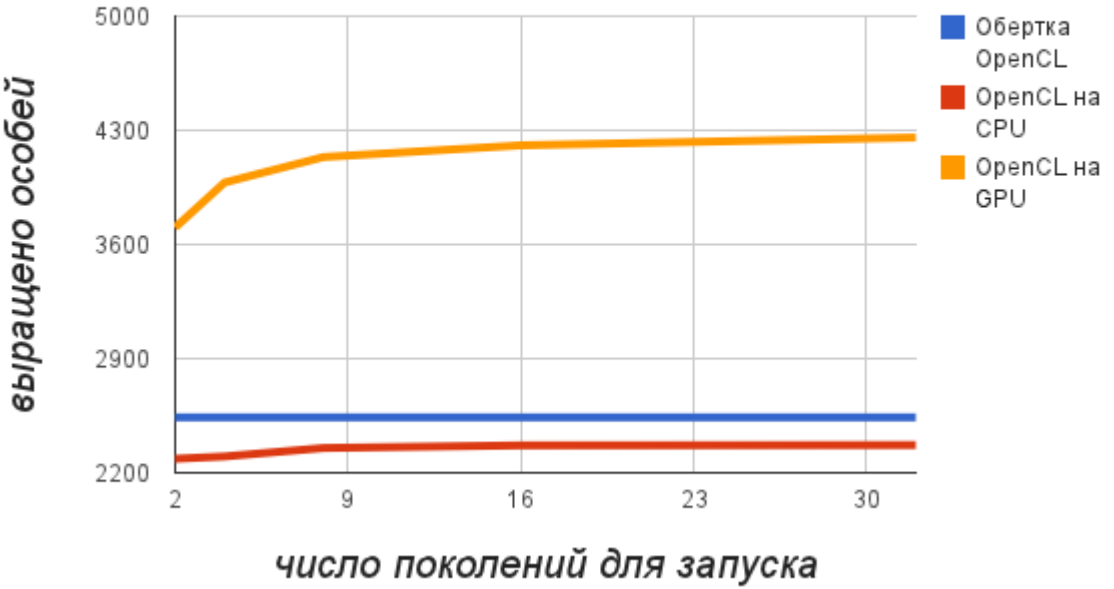
```
for ( uint i=0; i<bufSize; ++i )
{
    random_value = nextInt( random_value );
    uint res = select( myBuf, hisBuf, (random_value & 512) );
    tempBuffer[ myBuf+i ] = inBuffer[ res+i ];
}
```



```
for ( uint i=0; i<cnt; ++i )
{
    char16 father = as_char16(vload4( i, inBuffer+myBuf));
    char16 mother = as_char16(vload4( i, inBuffer+hisBuf));
    char16 res = select( father, mother, (r << 3) );
    r = nextUChar16( r );
    vstore4( as_uint4(res), i, tempBuffer+myBuf );
}
```

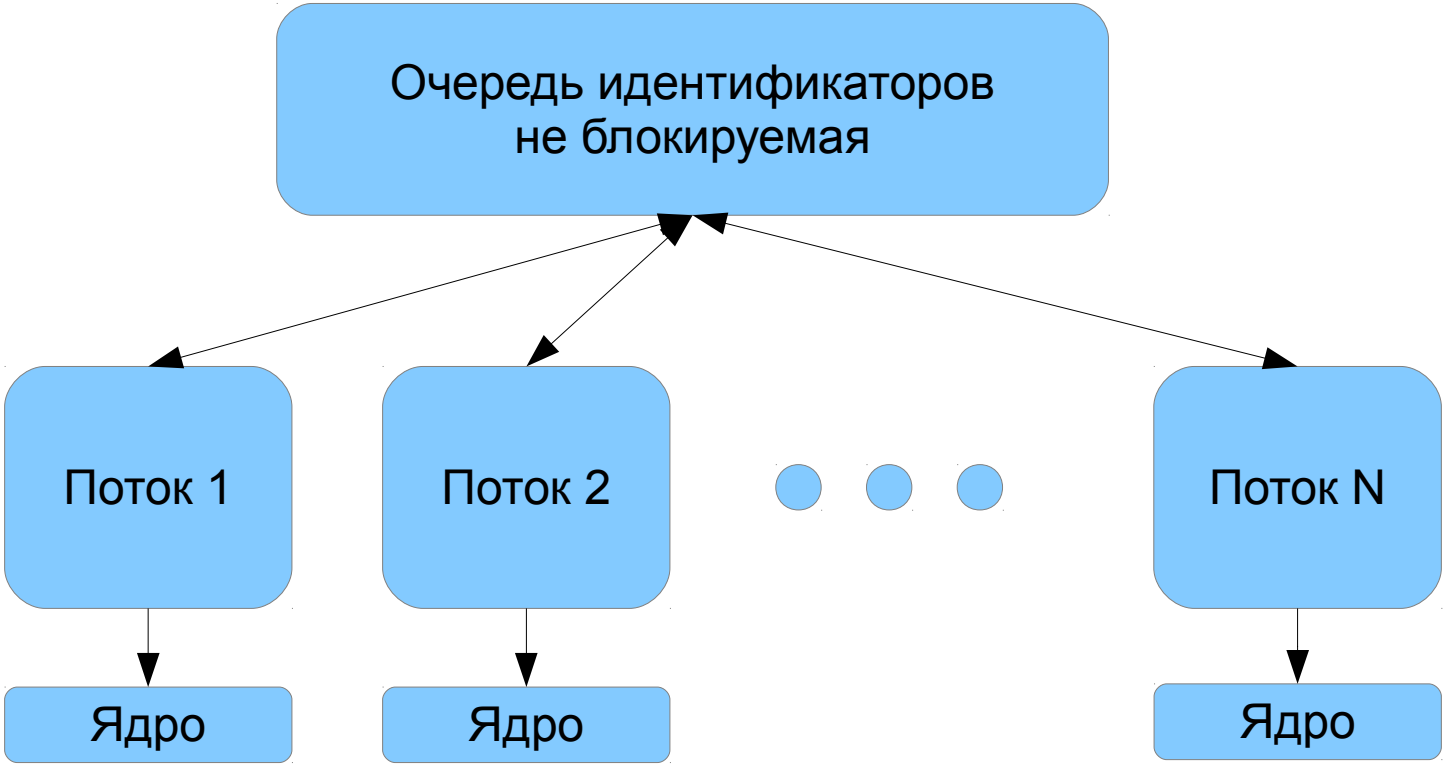
Другие особенности

- Для уменьшения влияния издержек на вызов ядра *OpenCL*, запуск производится сразу на 20 поколений



- Для уменьшения времени доступа к памяти, по возможности используется локальная память
- Реализовано наиболее эффективное копирование памяти

«Правильная многопоточность» (далее «Обертка над *OpenCL*»)



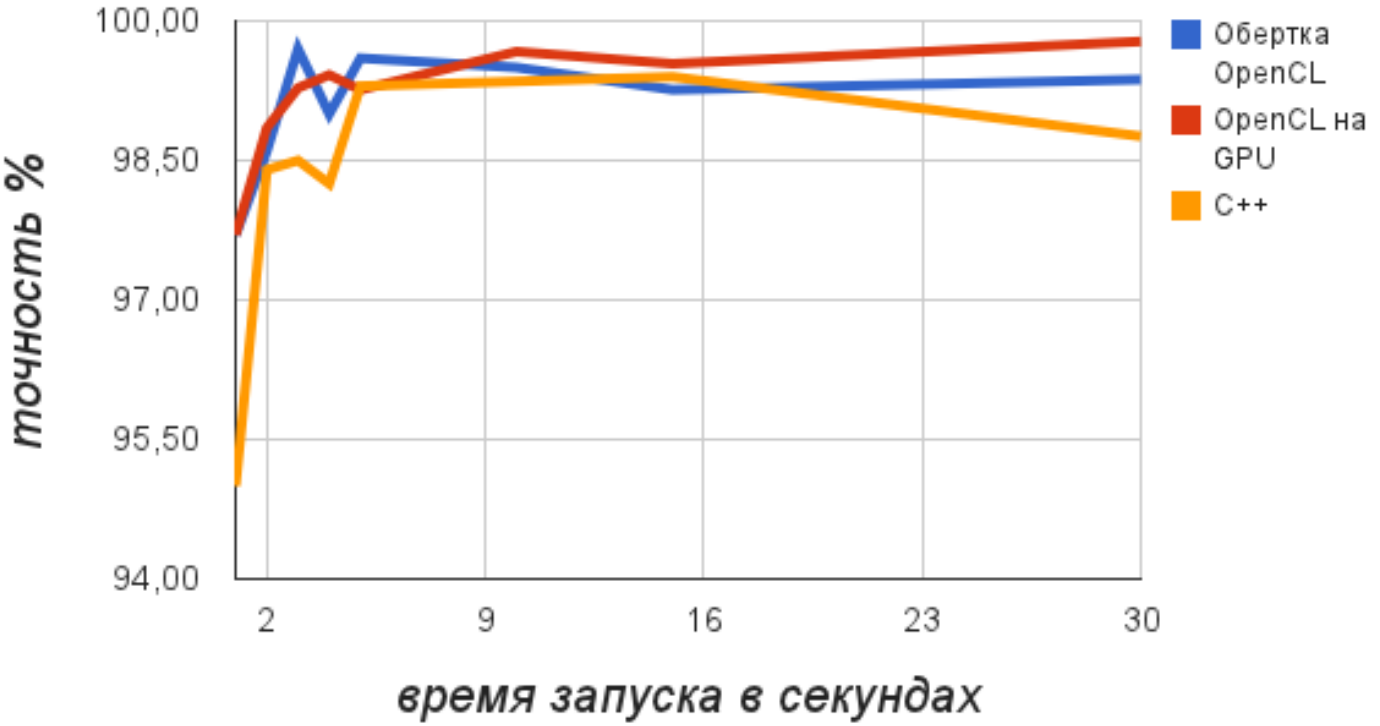
- В виду отсутствия функций, возвращающих идентификаторы потока и размеры группы, в ядро необходимо передавать специальную структуру с данными
- Векторные типы заменены на массивы

Измерения результатов

- Измерения проводились на различных машинах
- Измерялись четыре вида программ: *OpenCL* на GPU, *OpenCL* на CPU, Обертка над *OpenCL*, простая многопоточная версия на C++
- Проверялось несколько версий кода *OpenCL*
- Тестирования проводились с различными размерами поколений: от 25 особей до более чем 16 тысяч особей
- Метрикой результата считалось число поколений, выращенных за 30(либо 10) секунд выполнения программы

Достоверность результатов

- Экспериментальным путем установлено,
  что среднеквадратичная погрешность результата
  при выбранной метрике оценивания не превышает 2%,
  а при запуске на графическом процессоре этот показатель
  еще меньше



Результаты измерений для модификации задачи об умном муравье

- CPU: AMD Phenom II X4 955 3.20 GHz
- GPU: AMD Radeon HD 6850Ti

Размер поколения	Обертка OpenCL	OpenCL на GPU	OpenCL на CPU	C++
1024	7657	12620	7020	1032
3072	2611	5100	2320	337
4098	1960	3400	1760	268
16384	500	560	440	74

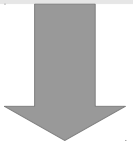
Число поколений, построенных за 30 секунд выполнения алгоритма

Результаты измерений

- CPU: AMD Phenom II X4 955 3.20 GHz
- GPU: AMD Radeon HD 6850Ti

Переход от скалярных операций мутации и скрещивания к векторным.

Размер поколения	OpenCL на GPU	OpenCL на CPU
25	19880	77440
256	17980	7680



Размер поколения	OpenCL на GPU	OpenCL на CPU
25	23020	71040
256	20160	7280

Число поколений, построенных за 30 секунд выполнения алгоритма

Переход на Задачу об Умном муравье-3

- Генетический алгоритм выполняется на центральном процессоре, на *OpenCL* рассчитываются только функции приспособленности
- Реализованы другие способы хранения информации о переходах, сокращенные таблицы и деревья решений
- Внесены необходимые изменения в виртуальную лабораторию

Результаты измерений для задачи об Умном муравье 3

- CPU: AMD Phenom II X4 955 3.20 GHz
- GPU: AMD Radeon HD 6850Ti

Размер поколения	OpenCL на GPU	OpenCL на CPU	C++
3072	782.50	734.50	288.75
16384	120.00	122.00	52.25

Среднее число поколений, построенное за 10 секунд работы алгоритма.

Заключение

- Решения, примененные в данной работе, могут быть использованы для решения других задач
- Возможно использовать мощности графического процессора в дополнение к центральному
- Платформа *OpenCL* является мощным инструментом, позволяющим производить эффективные вычисления на различных устройствах

Спасибо за внимание!

Вопросы?