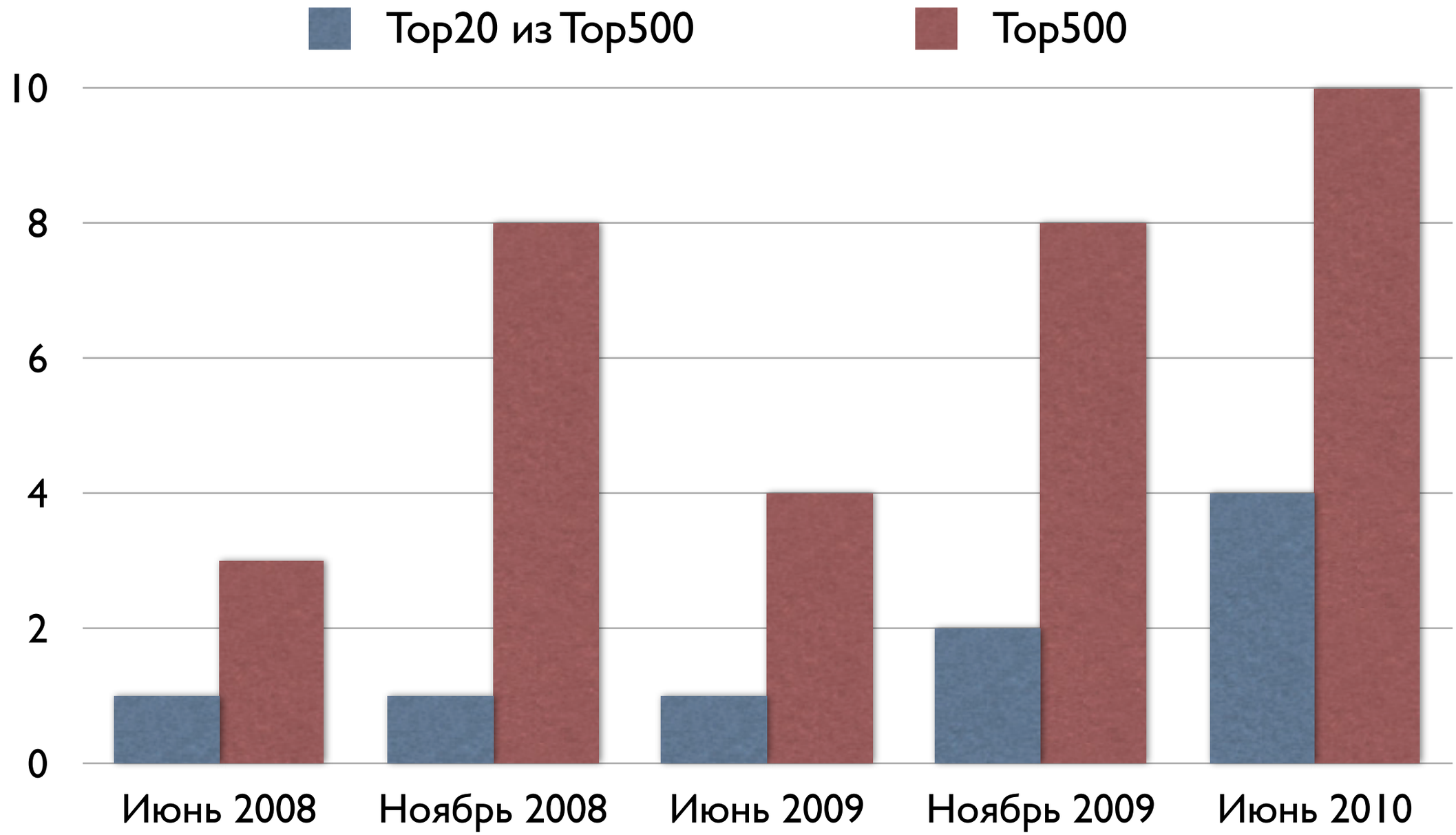
Технология программирования неоднородных параллельных систем OpenCL

Адинец Андрей Викторович,
к.ф.-м.н., мл.н.с. НИВЦ МГУ
adinetz@gmail.com

Ускорители в Top500



Как программировать

- Разные ускорители
- Разные языки и библиотеки
  - NVidia — CUDA
  - AMD — Brook+
  - CELL BE — SPU-C
  - x86 — C/C++/Fortran + OpenMP
- Непереносимость
  - Код
  - Навыки программиста

Цели OpenCL

- Стандарт программирования
  - Многоядерные процессоры
  - Ускорители, ГПУ
  - Мобильные медиапроцессоры
- Стандарт функциональности
  - Производителям процессоров

Рабочая группа OpenCL

Рабочая группа OpenCL

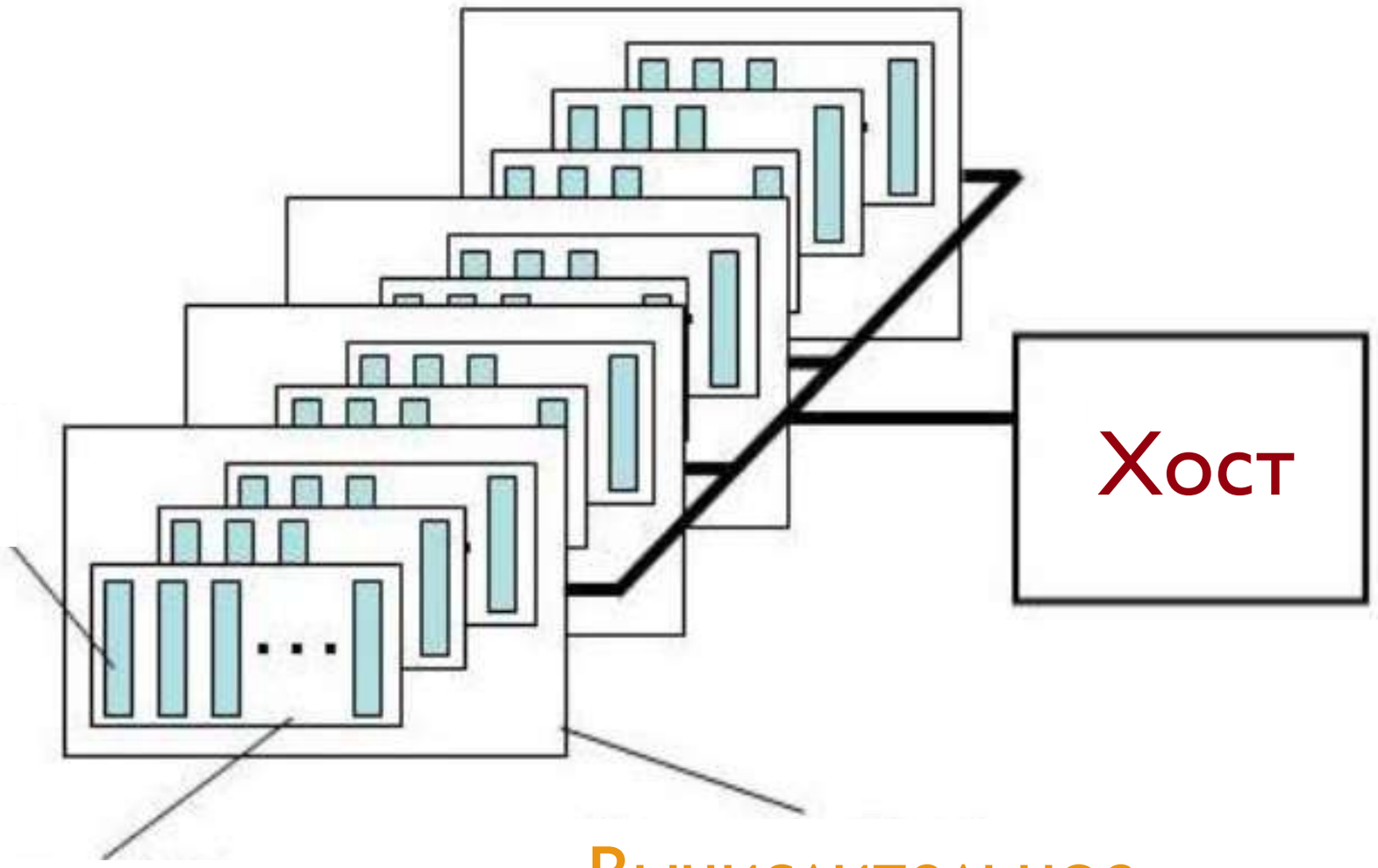
План

- Модель системы
- Среда исполнения
- Возможности оптимизации
- Расширения
- Заключение

План

- **Модель системы**
- Среда исполнения
- Возможности оптимизации
- Расширения
- Заключение

Модель системы

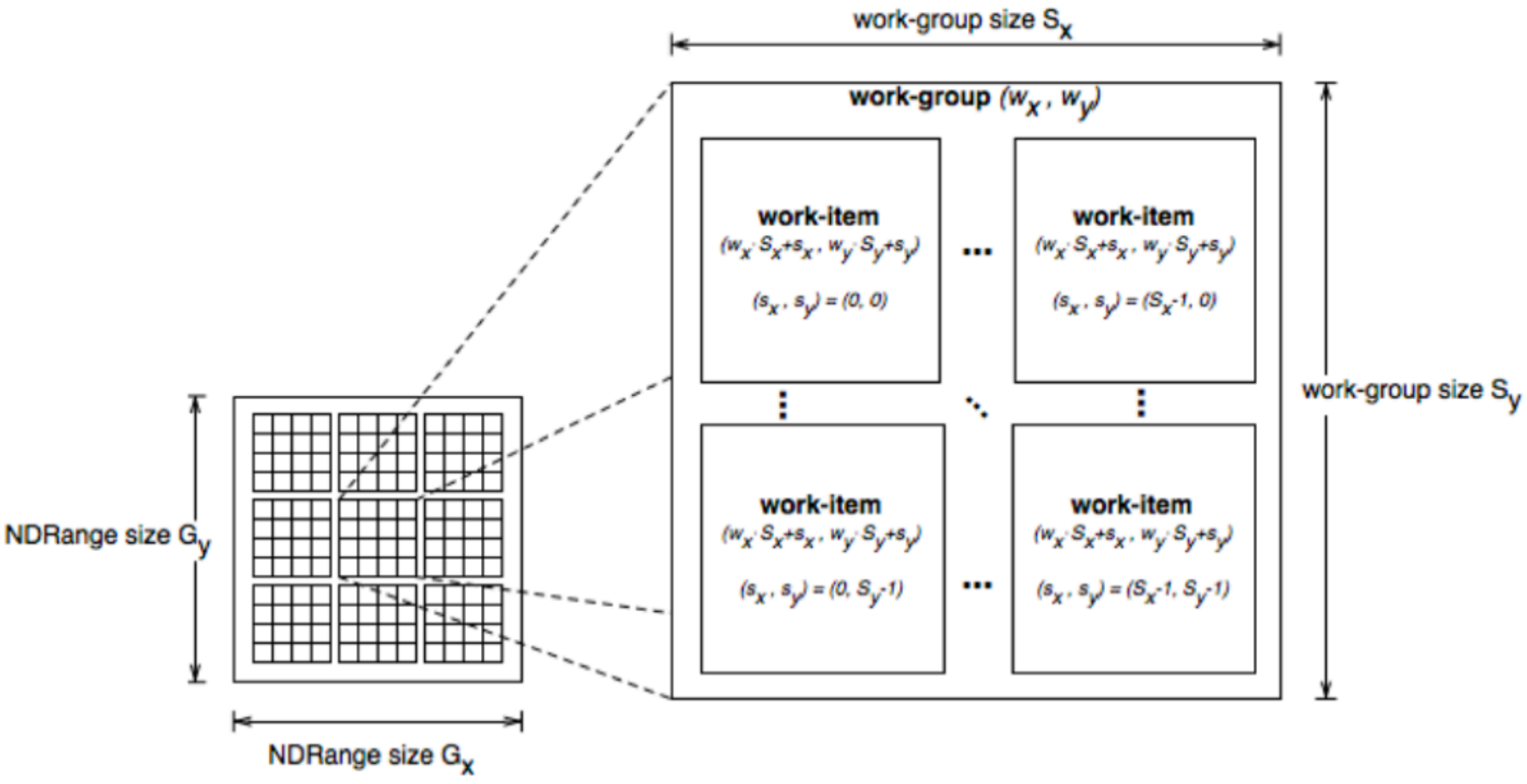


Модель программы

- Хост-программа
- Программа для устройства
- Набор ядер

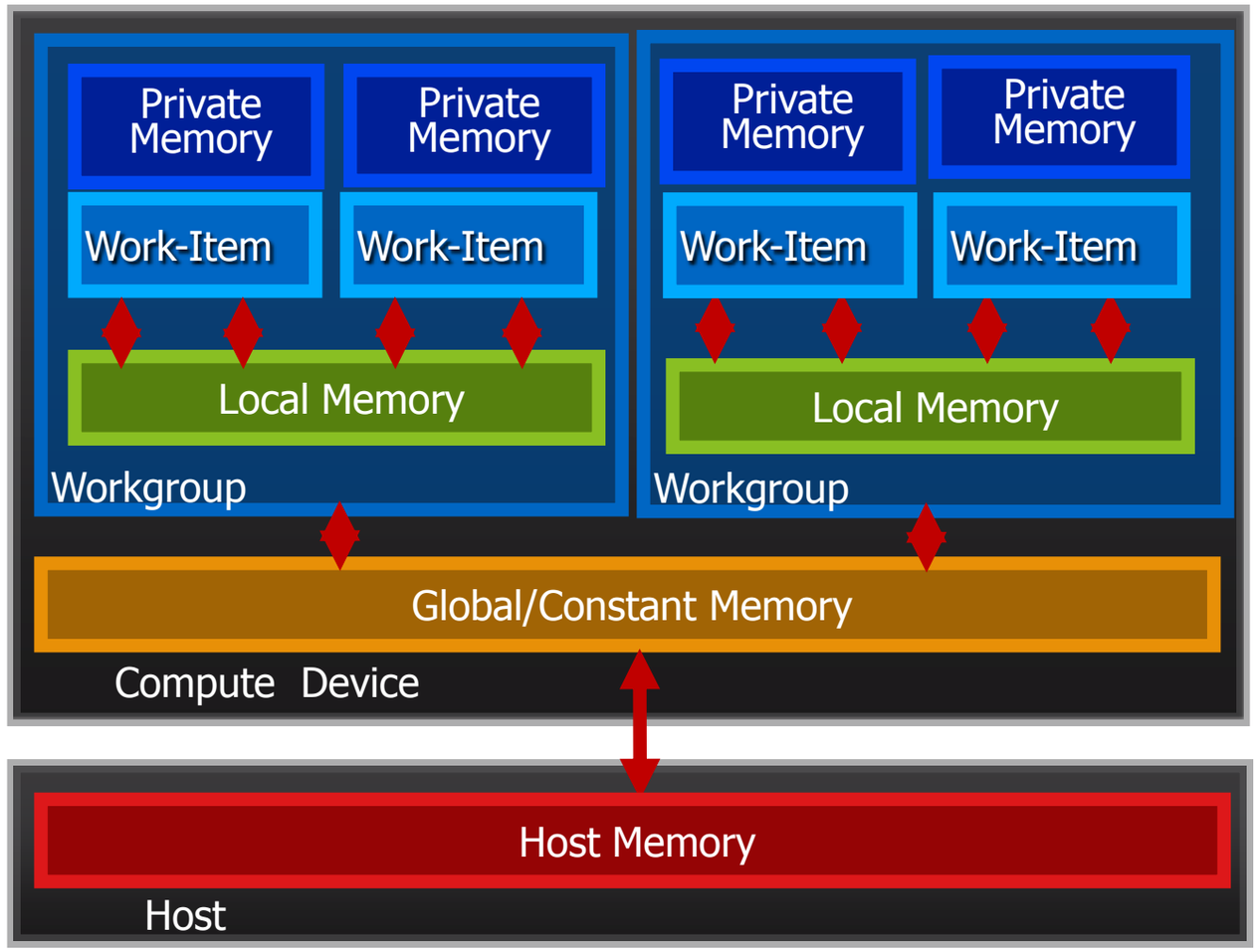
Аппаратура	Программа
Потоковый процессор	Поток
Мультипроцессор	Блок потоков
Устройство	Ядро
Хост	Хост-программа

Исполнение ядра



Иерархия памяти

- Приватная
- Локальная
- Константная
- Глобальная
- Хост-память



Соответствие иерархий

Аппаратура	Исполнение	Память	Модификатор
Потоковый процессор	Поток	Приватная	`private`, —
Мультипроцессор	Блок потоков	Локальная	`local`
Устройство	Ядро	Глобальная Константная	`global` `constant`
Хост	Хост-программа	Хост-память	—

Модификаторы

- либо `__xxx`, либо `xxx`
- Модификатор `kernel`
  - Функция является ядром
- Модификаторы классов памяти
  - `private`, `local`, `constant`, `global`

Где я?

- Размер блока и решётки в ПП
  - `get_local_size(int dim)`,`get_global_size(int dim)`
- Размер решётки в блоках
  - `get_num_groups(int dim)`
- Номер ПП в блоке и решётке
  - `get_local_id(int dim)`,`get_global_id(int dim)`
- Номер блока
  - `get_group_id(int dim)`

Пример ядра

```
kernel void addarr(
   global float* c,
   global float* a,
   global float* b) {
   int j = get_global_id(0);
   c[j] = a[j] + b[j];
}  // end of addarr()
```

Чего нет

- Указатель на функцию
- Рекурсия
- Нормальные указатели
  - `global * global p;` — нельзя!
- Стандартная библиотека C

План

- Модель системы
- **Среда исполнения**
- Возможности оптимизации
- Расширения
- Заключение

Библиотека OpenCL

- Заголовок
  - `#include <CL/cl.h>`
- Библиотека
  - `gcc -lOpenCL myprog.c -omyprog`

Платформа

- Конкретная реализация OpenCL
- Получить платформы
  - `clGetPlatformIDs()`
- Информация о платформе
  - `clGetPlatformInfo()`

Устройство

- Конкретное вычислительное устройство
- Получить устройства
  - `clGetDeviceIDs()`
- Получить информацию
  - `clGetDeviceInfo()`

Демонстрация

Информация об устройстве

Объекты библиотеки

- Создание
  - `cl_xxx clCreateXxxYyy(..., cl_int *err);`
- Счётчик ссылок
  - `+ : cl_int clRetainXxx(cl_xxx id);`
  - `– : cl_int clReleaseXxx(cl_xxx id);`
- Получить свойство
  - `cl_int clGetXxxInfo(cl_xxx id, cl_xxx_info param, size_t sz, void *val, size_t* sz_ret);`

Контекст и очередь

- Контекст
  - ~ процесс — память, программы
  - >= 1 однотипных устройств
- Очередь
  - ~ поток — исполнение команд
  - = 1 устройство

Команды

- Постановка в очередь
  - `clEnqueueXxx(cl_command_queue queue_id, ..., cl_uint nevents, const cl_event* wait_list, cl_event* ev);`
- Исполнение
  - `clFlush()` — отправить на устройство
  - `clFinish()` — дождаться завершения

Буфер данных

- Одномерный массив в памяти хоста или устройства
- Копирование
  - `clEnqueue{Read,Write,Copy}Buffer()`
  - Блокирующее/неблокирующее
- Отображение
  - `clEnqueue{Map,Unmap}Buffer()`

Программа

- Исполняемый код устройства
  - >= 1 ядер
- Создание
  - `clCreateProgramWith{Source,Binary}()`
- Сборка
  - `clBuildProgram()`
  - `clGetProgramBuildInfo()`

Ядро

- «Точка входа» в устройство
- Создание
  - `clCreateKernel()`, `clCreateKernelsInProgram()`
- Параметры
  - `clSetKernelArg()`
- Запуск
  - `clEnqueueNDRangeKernel()` — на решётке

Демонстрация

Hello, OpenCL World!

План

- Модель системы
- Среда исполнения
- **Возможности оптимизации**
- Расширения
- Заключение

Задача N тел

- N частиц + 4 закона Ньютона
- Рассчитать эволюцию системы

$$\mathbf{a}_i = \sum_{i=1}^{N} G \frac{m_j(\mathbf{r}_j - \mathbf{r}_i)}{\sqrt{|\mathbf{r}_j - \mathbf{r}_i|^2 + \epsilon^2}^3}$$

Профилировка

- `clCreateCommandQueue(ctx, dev, CL_QUEUE_PROFILING_ENABLE, 0);`
- `clEnqueueXxx(queue, ..., 0, 0, &event);`
- `clGetEventProfilingInfo(&event, name, sz, &value, 0);`
- `name = CL_PROFILING_COMMAND_{QUEUED, SUBMIT, START, END}`

Демонстрация

Задача N тел — простой вариант

От забора до барьера

- `CL_{LOCAL,GLOBAL}_MEM_FENCE`
- Забор
  - `mem_fence(cl_mem_fence_flags flags)`
  - `{read|write}_mem_fence (cl_mem_fence_flags flags)`
- Барьер
  - `barrier(cl_mem_fence_flags flags)`

Асинхронное копирование

- `event_t async_work_group_copy (gentype* dst, gentype* src, size_t n, event_t ev);`
- `void wait_group_events(int n, event_t *evs);`

Векторные типы

- `type{2,3,4,8,16}` — `uchar8`, `float4`
- Перестановки (swizzles)
  - `b = a.zzyy; b.wyzw = a; c.odd = b;`
- Арифметические операции
- Стандартные функции

Демонстрация

Задача N тел — оптимизации

План

- Модель системы
- Среда исполнения
- Возможности оптимизации
- **Расширения**
- Заключение

Расширения

- `cl_khr_<имя-расширения>`
- Определения
  - `enum CL_<имя-перечисления>_KHR`
  - `cl<имя-функции>KHR`
- Использовать в ядре:
  - `#pragma OPENCL EXTENSION cl_khr_<имя-расширения> : {enable|disable}`

Расширения (2)

- Взаимодействие с графикой:
  - cl_khr_gl_sharing, cl_khr_d3d10_sharing
- Новые типы данных:
  - cl_khr,_fp64, cl_amd_fp64, cl_khr_half
- Атомарные операции
  - cl_khr_{local|global}_int{32|64}_{base|extended}_atomics

Атомарные операции

- `T atom_op(T *p, T val)`
  - `T=int, unsigned int, long, unsigned long`
  - `op=add, sub, xchg, min, max, and, or, xor`
- `T atom_op(T *p)`
  - `op=inc, dec`
- `T atom_cmpxchg(T *p, T cmp, T val)`
- Атомарность
  - «Одновременно» исполняет только один поток

Задача: гистограмма

```
for(int i = 0; i < n; i++) {
  bins[data[i]]++;
}
```

```
kernel void histogram(global int
*bins, global uchar* data, int n) {
  int i = get_global_id(0);
  atom_inc(&bins[data[i]]);
}
```

Демонстрация

Гистограмма

План

- Модель системы
- Среда исполнения
- Возможности оптимизации
- Расширения
- **Заключение**

Чего не хватает

- Один код для хоста и ускорителя
- Нормальные указатели
- Производительность

Практикум

- 16:40–18:10, М3-2, 10 x NVidia Tesla
- Дано: «болванка» проекта на C + OpenCL
- Задания
  - w[i] = a * x[i] + b * y[i] * z[i]
  - умножение комплексных матриц в двойной точности -> max perf
  - ...

Ссылки

- http://www.khronos.org/opencl/
- http://developer.nvidia.com/object/opencl.html
- http://developer.amd.com/gpu/atistreamsdk/pages/default.aspx
- http://www.alphaworks.ibm.com/tech/opencl