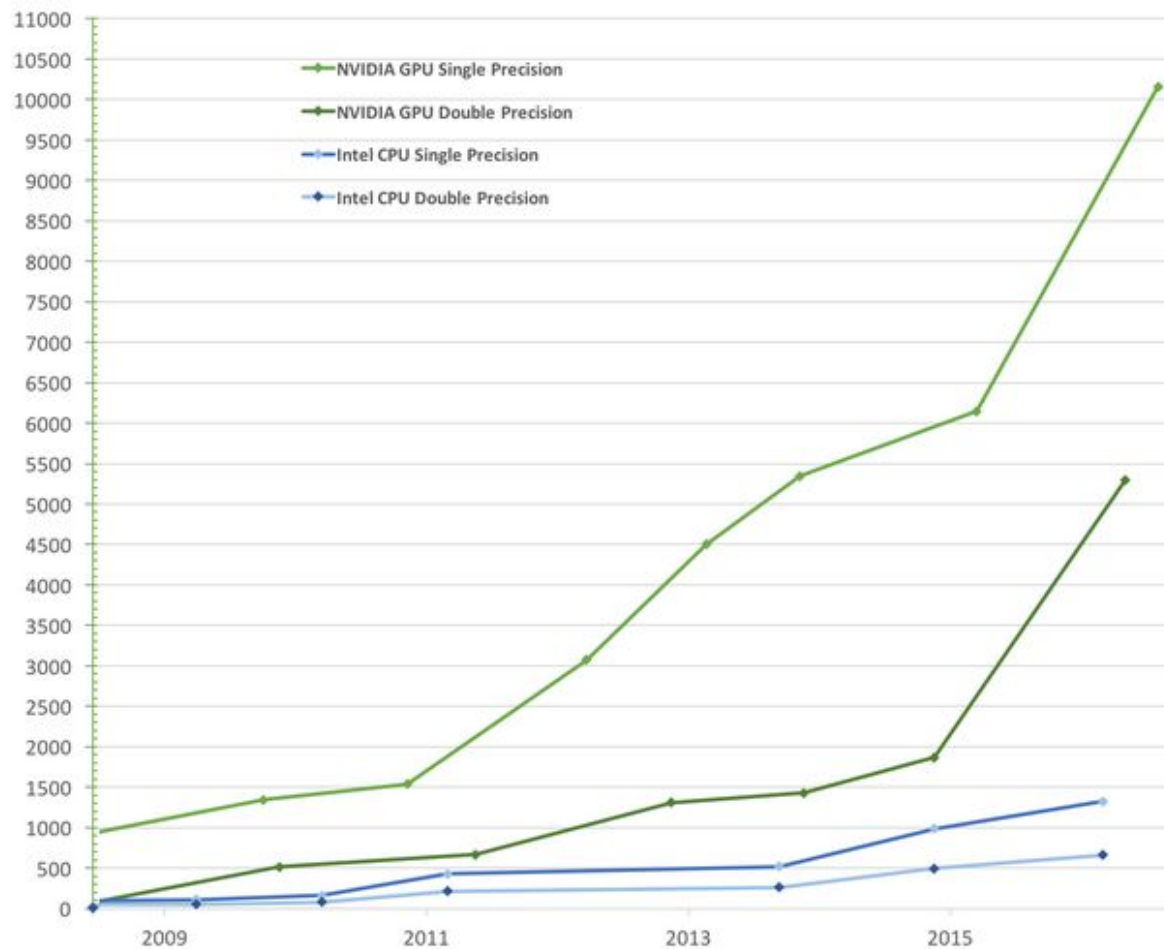


Лекция 1

- модели параллельных вычислений;
- аппаратные особенности графических процессоров;
- архитектура CUDA – основные свойства и принципы;
- программная модель: хост, устройства, ядра, иерархия нитей (threads);
- программный интерфейс CUDA;
- установка CUDA Toolkit.

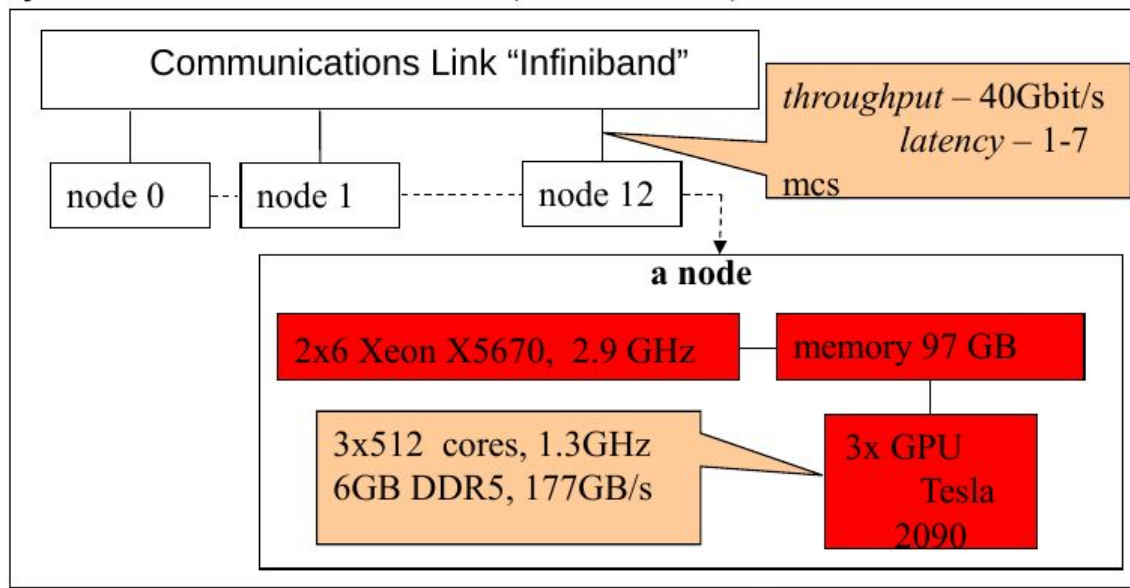
Theoretical GFLOP/s at base clock



Гибридный кластер НГУ (<http://nusc.ru>)

Кластер состоит из 12 узлов *HP SL390s G7*, каждый из которых содержит два 6-ядерных CPU *Xeon X5670* и три графические карты *NVIDIA Tesla M2090*.

Каждый GPU имеет **512 ядер** (cores) с частотой 1.3GHz и память размером **6GB** с пропускной способностью (bandwidth) 177 GB/s.



58 вакансий «cuda»

По соответствию ▾

За всё время ▾



На карте



Подработка

- ☐ От 4 часов в день 1
- ☐ По вечерам 1
- ☐ Неполный день
- ☐ Разовое задание
- ☐ Временная работа
- ☐ По выходным

Исключить слова

Исключить слова, через зап.

Уровень дохода



- ☒ Не имеет значения
- ☐ от 195 000 ₽ 12
- ☐ от 265 000 ₽ 5
- ☐ от 335 000 ₽ 4
- ☐ от 410 000 ₽ 2
- ☐ от 555 000 ₽ 1

Senior C++ Разработчик (Computer Vision)

от 300 000 ₽

НЕЙРОВИЖН
Москва

Опыт более 6 лет

Откликнуться

Разработчик C++ \ математик-алгоритмист (computer vision, 3D)

150 000 – 200 000 ₽

РэнджВижн

Нижний Новгород, Ленинская

Опыт от 3 до 6 лет

Откликнитесь среди первых


Откликнуться

Показать контакты

1 вакансия «cuda»

По соответствию ▾

За всё время ▾

 На карте



Подработка

- ☐ От 4 часов в день
- ☐ По вечерам
- ☐ Неполный день
- ☐ Разовое задание
- ☐ Временная работа
- ☐ По выходным

Исключить слова

Исключить слова, через зап.

Уровень дохода



- ☒ Не имеет значения
- ☐ от 200 000 ₽ 1
- ☐ Своя зарплата

от

По вашему запросу ещё будут появляться новые вакансии.
Присылать вам?

cuda • Новосибирская область • Ключевые слова в названии вакансии и в описании вакансии

Да, на почту


В мессенджер

Разработчик C++ Middle/Senior

130 000 – 200 000 ₽

Рекрутинговое агентство The One ✓

Новосибирск

 Опыт от 3 до 6 лет

Откликнитесь среди первых

Работодатель сейчас онлайн

Откликнуться

<https://developer.nvidia.com/cuda-zone>

<https://developer.nvidia.com/cuda-downloads>

https://github.com/mlkv52git/sibsutis_cuda_bachelors-2023

Модели параллельных вычислений

Модель	Программные средства	Архитектура ВС
Общая память	POSIX (pthread), WinAPI(CreateThread), OpenMP...	MIMD, разделяемая память
Обмен сообщениями	MPI (Message Passing Interface): OpenMPI, MPICH, LAM (Local Area Multicomputer); PVM (Parallel Virtual Machine)...	MIMD, распределенная и разделяемая память
Параллелизм данных	Языки .NET, Python...	MIMD/SIMD

Архитектура фон Неймана



Регистры общего назначения – сумматор, регистр данных, адресный регистр и т.д.

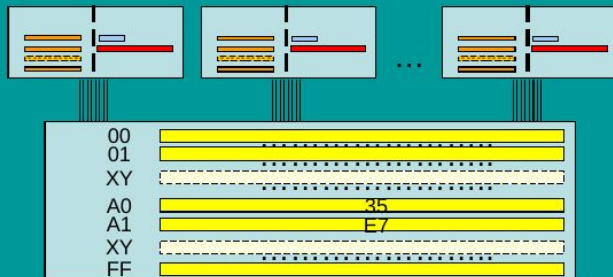
Счетчик команд

Ячейки памяти

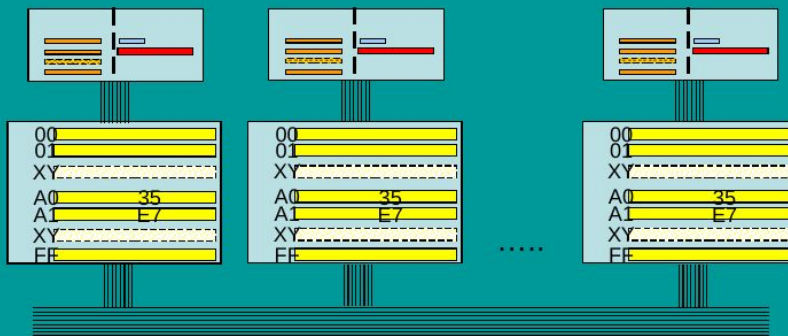
Регистр команд

Основные архитектуры производственных ВС

Разделяемая память

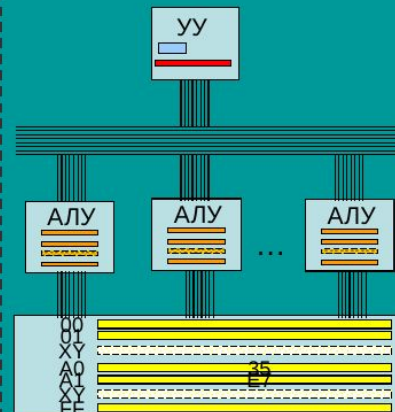


Распределенная память



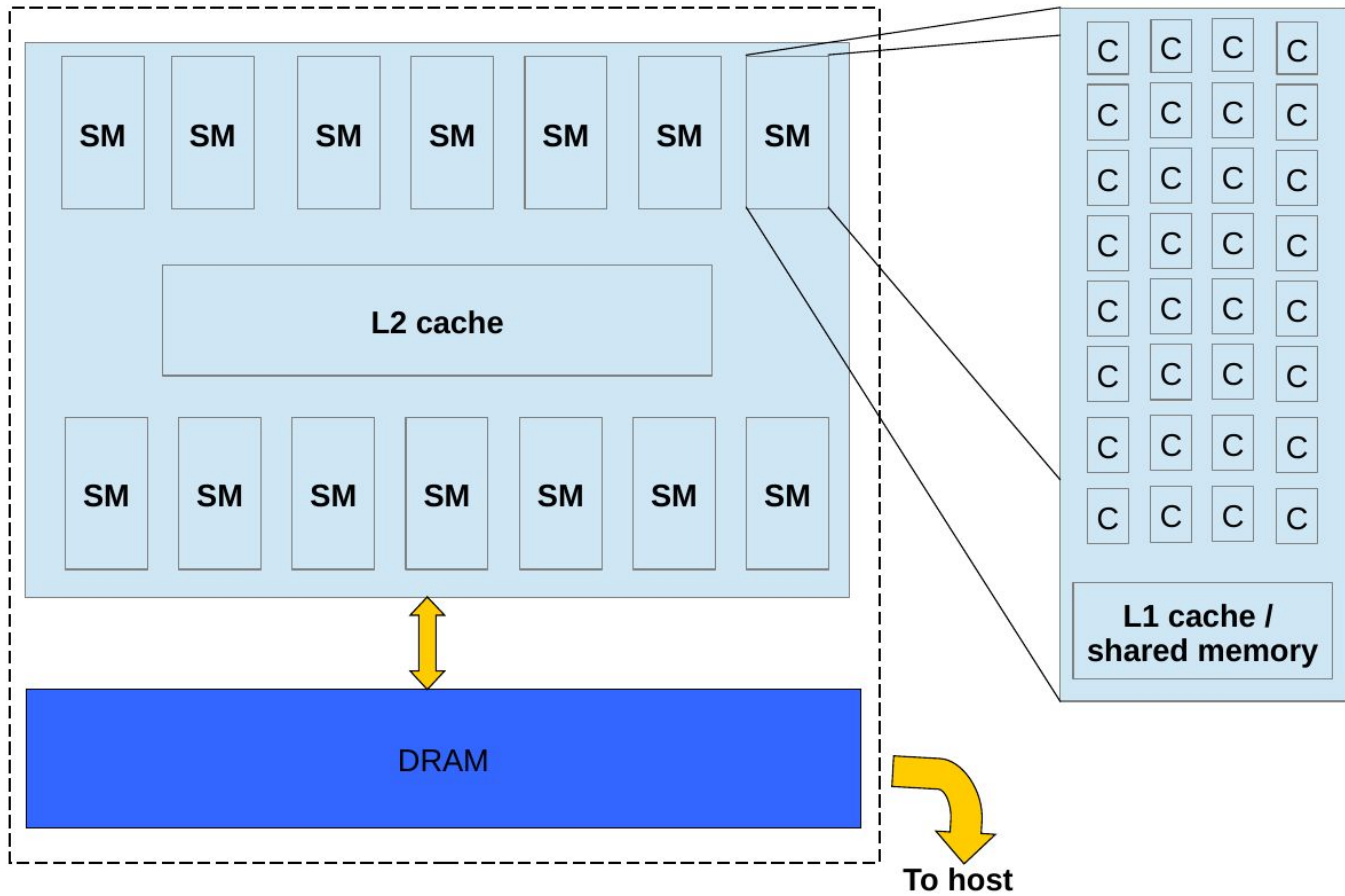
MIMD

Разделяемая память



SIMD

GPU (Fermi architecture)



SM



L0 Instruction Cache

Warp Scheduler (32 thread/clock)

Dispatch Unit (32 thread/clock)

Register File (16,384 x 32-bit)

FP64

INT

INT

FP32

FP32

FP64

INT

INT

FP32

FP32

FP64

INT

INT

FP32

FP32

FP64

INT

INT

FP32

FP32

FP64

INT

INT

FP32

FP32

FP64

INT

INT

FP32

FP32

FP64

INT

INT

FP32

FP32

FP64

INT

INT

FP32

FP32

TENSOR
CORE

TENSOR
CORE

LD/
ST

LD/
ST

LD/
ST

LD/
ST

LD/
ST

LD/
ST

LD/
ST

LD/
ST

SFU

Логическое представление GPU

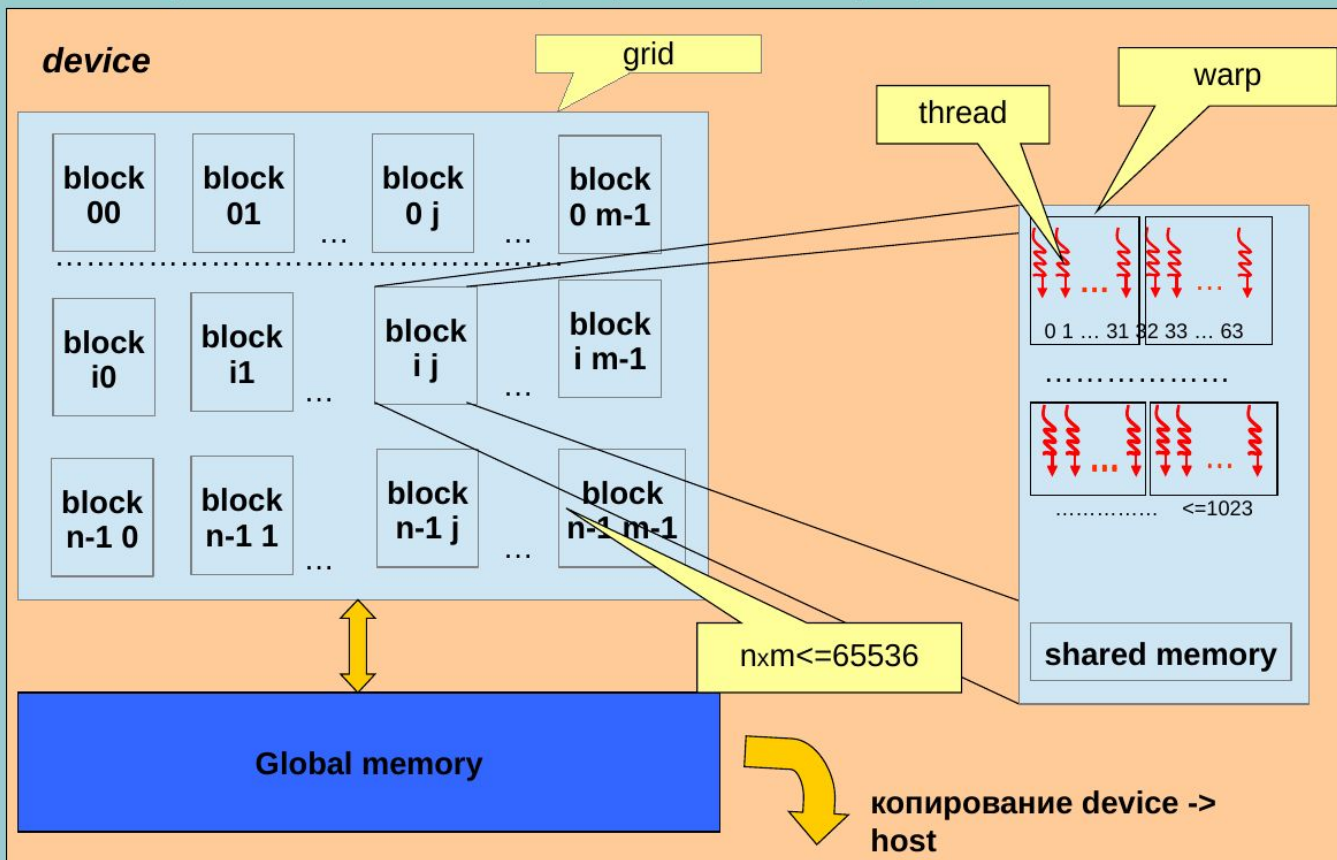
Активное использование графических процессоров (GPU) для прикладных расчетов научно-технического назначения во многом связано с предоставлением компанией NVIDIA технологии CUDA (*Compute Unified Device Architecture*). Технология CUDA предоставляет понятную для прикладного программиста абстракцию графического процессора (GPU) и простой интерфейс прикладного программирования (API – Application Programming Interface).

По терминологии CUDA вычислительный узел с CPU и main memory называется **host**, GPU называется **device**.

Программа, выполняемая на host'e содержит код – ядро (**kernel**), который загружается на device в виде многочисленных копий. Все копии загруженного кода – нити (**threads**), объединяются в блоки (**blocks**) по 512-1024 нити в каждом. Все блоки объединяются в сеть (**grid**) с максимальным количеством блоков 65536. Все нити имеют совместный доступ на запись/чтение к памяти большого объема - **global memory**, на чтение к кэшируемым **constant memory** и **texture memory**. Нити одного блока имеют доступ к быстрой памяти небольшого объема – **shared memory**.

CUDA (Compute Unified Device Architecture)

- cuda предоставляет абстракцию GPU для программистов



Расширение языка C **CUDA C** — спецификаторы функций и переменных, специальные директивы, встроенные переменные и новые типы данных, а так же набор функций и структур данных **CUDA API**, предоставляют простой инструмент для программирования на GPU.

Функция-ядро (kernel)

Код, выполняемый на устройстве (ядро), определяется в виде функции типа *void* со спецификатором **__global__**:

```
__global__ void gFunc(<params>){...}
```


Конфигурация нитей

При вызове ядра программист определяет количество нитей в блоке и количество блоков в *grid*. При этом допустима линейная, двумерная или трехмерная индексация нитей:

```
gFunc<<<dim3(bl_xdim, bl_ydim, bl_zdim),  
          dim3(th_xdim, th_ydim, th_zdim)>>>(<params>);
```

```
#include <stdio.h>
```

```
#include <cuda.h>
```

test.cu

```
__global__ void gTest(float* a){  
    a[threadIdx.x+blockDim.x*blockIdx.x]=  
        (float)(threadIdx.x+blockDim.x*blockIdx.x);  
}
```

```
int main(){  
    float *da, *ha;  
    int num_of_blocks=10, threads_per_block=64;  
    int N=num_of_blocks*threads_per_block;  
  
    ha=(float*)calloc(N, sizeof(float));  
    cudaMalloc((void**)&da, N*sizeof(float));
```

```
gTest<<<dim3(num_of_blocks),  
        dim3(threads_per_block)>>>(da);  
CudaDeviceSynchronize();  
  
cudaMemcpy(ha,da,N*sizeof(float),  
           cudaMemcpyDeviceToHost);
```

```
for(int i=0;i<N;i++)  
    printf("%g\n", ha[i]);
```

```
free(ha);  
cudaFree(da);
```

```
return 0;  
}
```

```
> nvcc test.cu -o test  
> ./test
```

<https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html>

rpm-пакет

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows	Linux	Mac OS X			
Architecture ¹	x86_64	ppc64le				
Distribution	Fedora	OpenSUSE	RHEL	CentOS	SLES	SteamOS
	Ubuntu					
Version	13.2					
Installer Type ¹	runfile (local)	rpm (local)	rpm (network)			

Download Target Installer for Linux OpenSUSE 13.2 x86_64

cuda-repo-opensuse132-7-5-local-7.5-18.x86_64.rpm (md5sum: e6708f4b38cc9ed546d3a047c731698)

Download (1.1 GB)

Installation Instructions:

1. `sudo rpm -i cuda-repo-opensuse132-7-5-local-7.5-18.x86_64.rpm`
2. `sudo zypper refresh`
3. `sudo zypper install cuda`

For further information, see the [Installation Guide for Linux](#) and the [CUDA Quick Start Guide](#).

После установки драйвера: `sudo nvidia-xconfig`

run-файл

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows	Linux	Mac OS X			
Architecture ¹	x86_64	ppc64le				
Distribution	Fedora	OpenSUSE	RHEL	CentOS	SLES	SteamOS
	Ubuntu					
Version	13.2					
Installer Type ¹	runfile (local)	rpm (local)	rpm (network)			

Download Target Installer for Linux OpenSUSE 13.2 x86_64

cuda_7.5.18_linux.run (md5sum: 4b3bcecf0dc35928a0898793cf3e4c6)

Download (1.1 GB)

Installation Instructions:

1. Run `sudo sh cuda_7.5.18_linux.run`
2. Follow the command-line prompts

The GPU Deployment Kit is available as a separate download [here](#).

For further information, see the [Installation Guide for Linux](#) and the [CUDA Quick Start Guide](#).

1. Выгрузить X-server (`sudo init 3`)
2. Блокировать загрузку драйвера **nouveau**