CUDA-библиотеки

План

- cuBLAS
- cuFFT
- cuSPARSE
- Thrust
- NPP
- cuRAND
- cuDNN
- и другие

BLAS - Basic Linear Algebra Subprograms

- Основа всех операций линейной алгебры
- Существенно зависит от архитектуры
- Множество остальных библиотек используют BLAS
- Три уровня функций:

  Level 1- операции с векторами $\mathbf{y} \leftarrow \alpha\mathbf{x} + \mathbf{y}$

  Level 2- Операции с матрицей и вектором $\mathbf{y} \leftarrow \alpha A\mathbf{x} + \beta\mathbf{y}$

  Level 3- Операции с несколькими матрицами $C \leftarrow \alpha AB + \beta C$

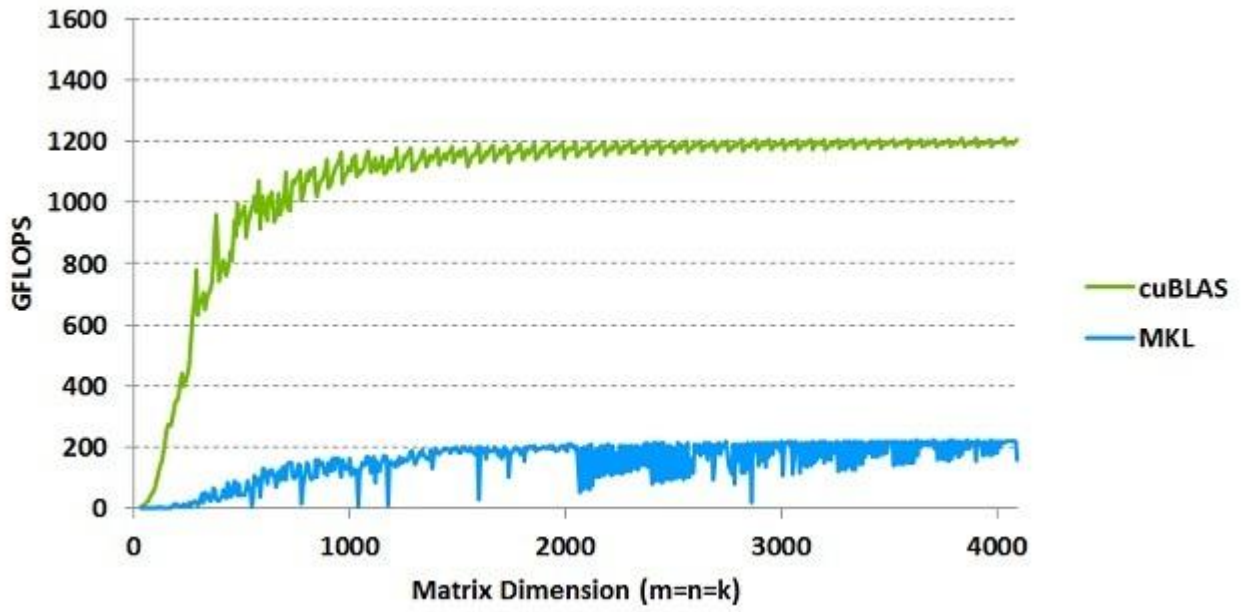
- **cuBLAS-XT** - использование нескольких видеокарт

BLAS. Имена функций

- **Шаблон: <character> <name> <mod>**
- **<character>** - тип компонент аргументов: float – s; double – d; cuComplex – c; cuDoubleComplex – z
- **<name>** - в Level 1 имя функции: max, min, sum, axpy, dot, …, в Level 2,3 – тип матрицы: ge - general matrix, gb - general band matrix, sy - symmetric matrix, tr - triangular matrix, …
- **<mod>** - в Level 1 - модификация аргументов, в Level 2,3 – имя функции mv – A*x, mm – A*B, sv – решение СЛАУ, …

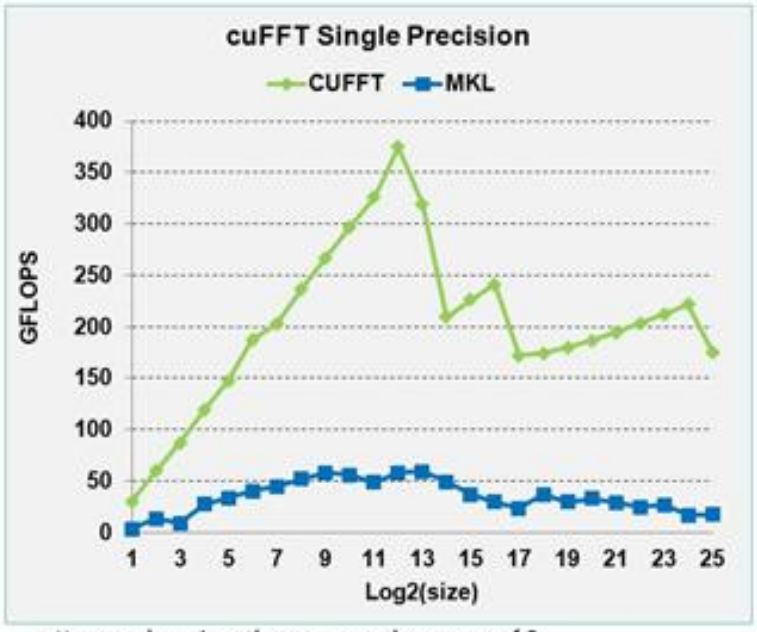
BLAS. Производительность

cuBLAS on K40m, ECC ON, input and output data on device.
MKL 11.0.4 on Intel IvyBridge single socket 12 -core E5-2697 v2 @ 2.70GHz

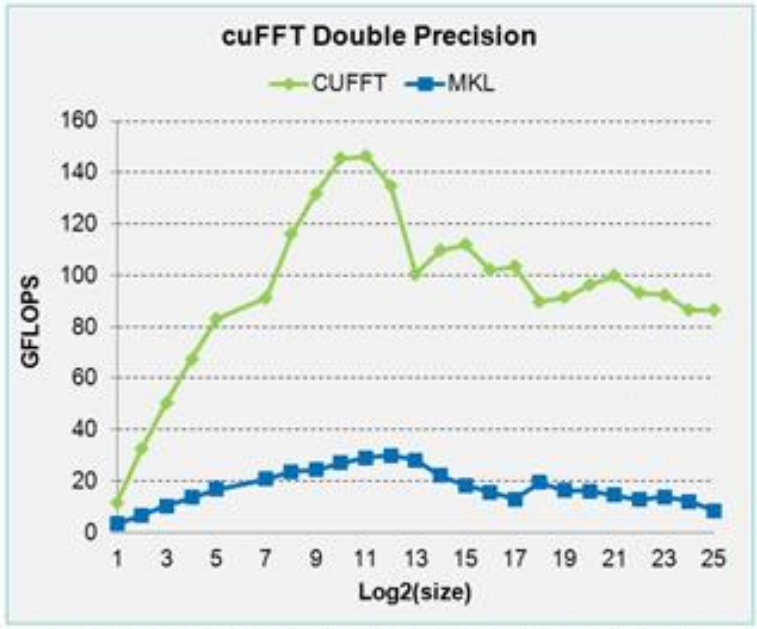


cuFFT - БПФ

- 1d,2d,3d, float/double, пакетные операции

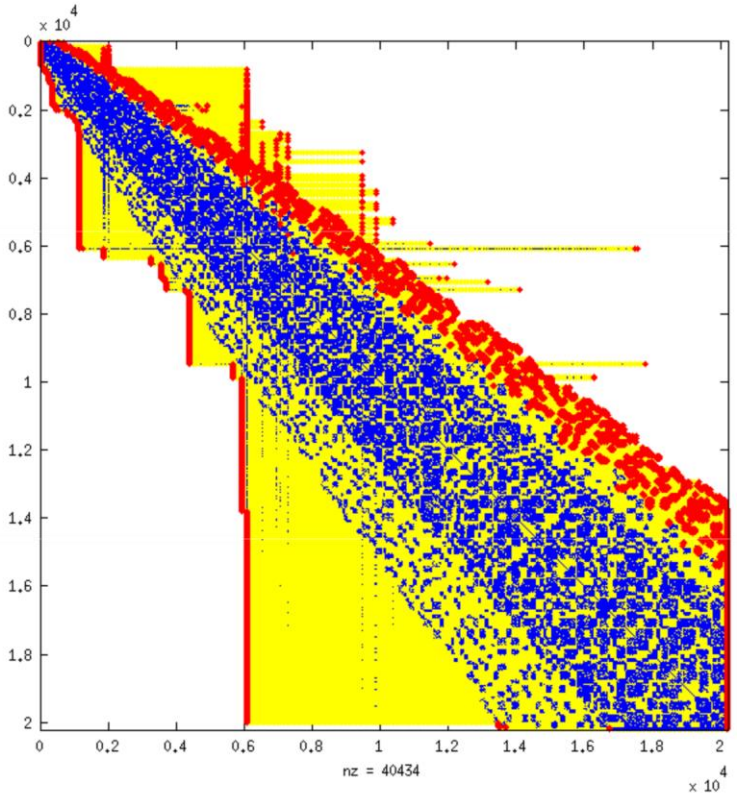


- Measured on sizes that are exactly powers-of-2
- cuFFT 4.1 on Tesla M2090, ECC on
- MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz



- MKL 10.2.3, TYAN FT72-B7015 Xeon x5680 Six-Core @ 3.33 GHz
- Performance may vary based on OS version and motherboard configuration

Разреженные матрицы



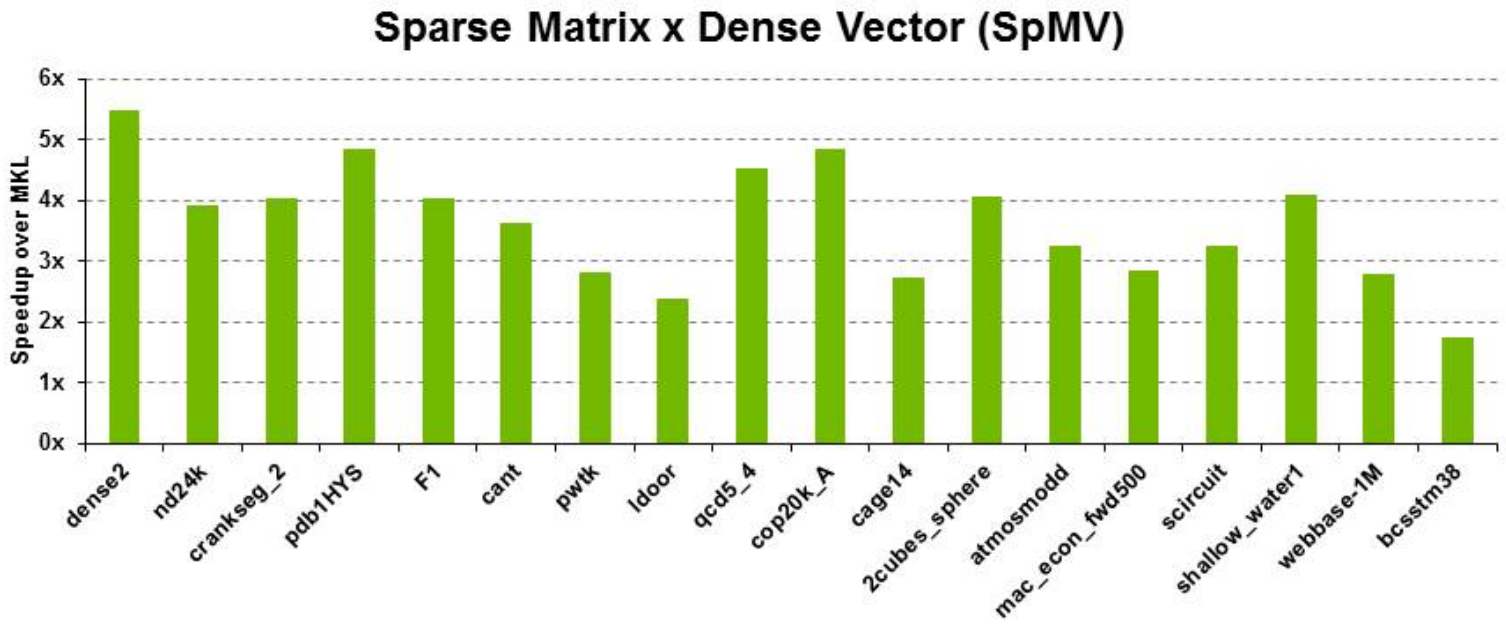
- Пример взят из пакета FDMNES решения уравнения Шрёдингера
- Другие приложения: моделирование (метод конечных элементов решения ДУ)
- Информационный поиск
- Графовые задачи
- …

cuSPARSE - обработка разреженных матриц

- Архитектура повторяет 3х-уровневую организацию BLAS + преобразование форматов матриц
- Прямой решатель СЛАУ
- Форматы хранения разреженных матриц:
  - Coordinate Format (COO). 3 массива: значения ненулевых элементов (порядок - по строкам), их номера строк и их номера столбцов
  - Ellpack-Itpack Format (ELL) – хранит все в 2х матрицах mxk, где k – максимально число ненулевых элементов в строке

cuSPARSE. Производительность

cuSPARSE: 5x Faster than MKL



- Average of s/c/d/z routines
- cuSPARSE 6.0 on K40m, ECC ON, input and output data on device
- MKL 11.0.4 on Intel IvyBridge 12-core E5-2697 v2 @ 2.70GHz
- Matrices obtained from: http://www.cise.ufl.edu/research/sparse/matrices/

Performance may vary based on OS version and motherboard configuration

Thrust – CUDA C++ STL

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/sort.h>
#include <thrust/copy.h>
#include <cstdlib>

int main(void)
{
    // generate 32M random numbers on the host
    thrust::host_vector<int> h_vec(32 << 20);
    thrust::generate(h_vec.begin(), h_vec.end(), rand);

    // transfer data to the device
    thrust::device_vector<int> d_vec = h_vec;

    // sort data on the device (846M keys per second on GeForce GTX 480)
    thrust::sort(d_vec.begin(), d_vec.end());

    // transfer data back to host
    thrust::copy(d_vec.begin(), d_vec.end(), h_vec.begin());

    return 0;
}
```

Thrust – CUDA C++ STL

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/reduce.h>
#include <thrust/functional.h>
#include <cstdlib>

int main(void)
{
  // generate random data on the host
  thrust::host_vector<int> h_vec(100);
  thrust::generate(h_vec.begin(), h_vec.end(), rand);

  // transfer to device and compute sum
  thrust::device_vector<int> d_vec = h_vec;
  int x = thrust::reduce(d_vec.begin(), d_vec.end(), 0, thrust::plus<int>());
  return 0;
}
```

NPP: NVIDIA Performance Primitives

- Множество функций обработки изображений: преобразование цвета, геометрии, форматов, фильтры, цветовая статистика, JPEG-операции, …
- Пример: GrabCut






cuRAND – генерация случайных чисел. Пример

```
int main(void)
{
  // use 30K independent seeds
  int M = 30000;

  float estimate = thrust::transform_reduce(thrust::counting_iterator<int>(0),
                                            thrust::counting_iterator<int>(M),
                                            estimate_pi(),
                                            0.0f,
                                            thrust::plus<float>());
  estimate /= M;

  std::cout << std::setprecision(3);
  std::cout << "pi is approximately " << estimate << std::endl;

  return 0;
}
```

cuRAND – генерация случайных чисел. Пример

```
struct estimate_pi : public thrust::unary_function<unsigned int,float>
{
  __host__ __device__
  float operator()(unsigned int thread_id)
  {
    float sum = 0;
    unsigned int N = 10000; // samples per thread

    unsigned int seed = hash(thread_id);

    // seed a random number generator
    thrust::default_random_engine rng(seed);

    // create a mapping from random numbers to [0,1)
    thrust::uniform_real_distribution<float> u01(0,1);
```

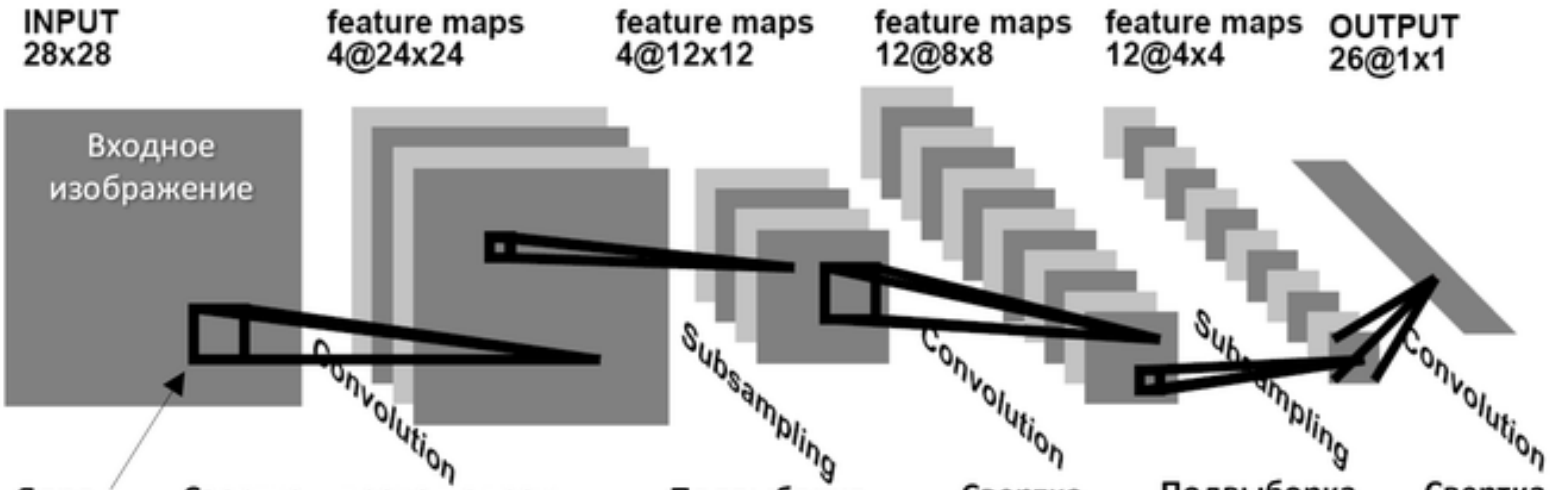
cuRAND – генерация случайных чисел. Пример

```
    // take N samples in a quarter circle
    for(unsigned int i = 0; i < N; ++i)
    {
      // draw a sample from the unit square
      float x = u01(rng);
      float y = u01(rng);

      // measure distance from the origin
      float dist = sqrtf(x*x + y*y);

      // add 1.0f if (u0,u1) is inside the quarter circle
      if(dist <= 1.0f)
        sum += 1.0f;
    }

    // multiply by 4 to get the area of the whole circle
    sum *= 4.0f;

    // divide by N
    return sum / N;
  }
};
```

Сверточные нейронные сети – глубокое обучение

- Используют некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток

Сверточные нейронные сети – глубокое обучение



Ядро свертки – матрица из весов, например, размером 5x5

Свертка – проход ядром свертки по всем нейронам нижнего слоя для активации нейронов верхнего слоя – получение карты признаков (feature maps). Для ядер разного типа получаются разные карты признаков.

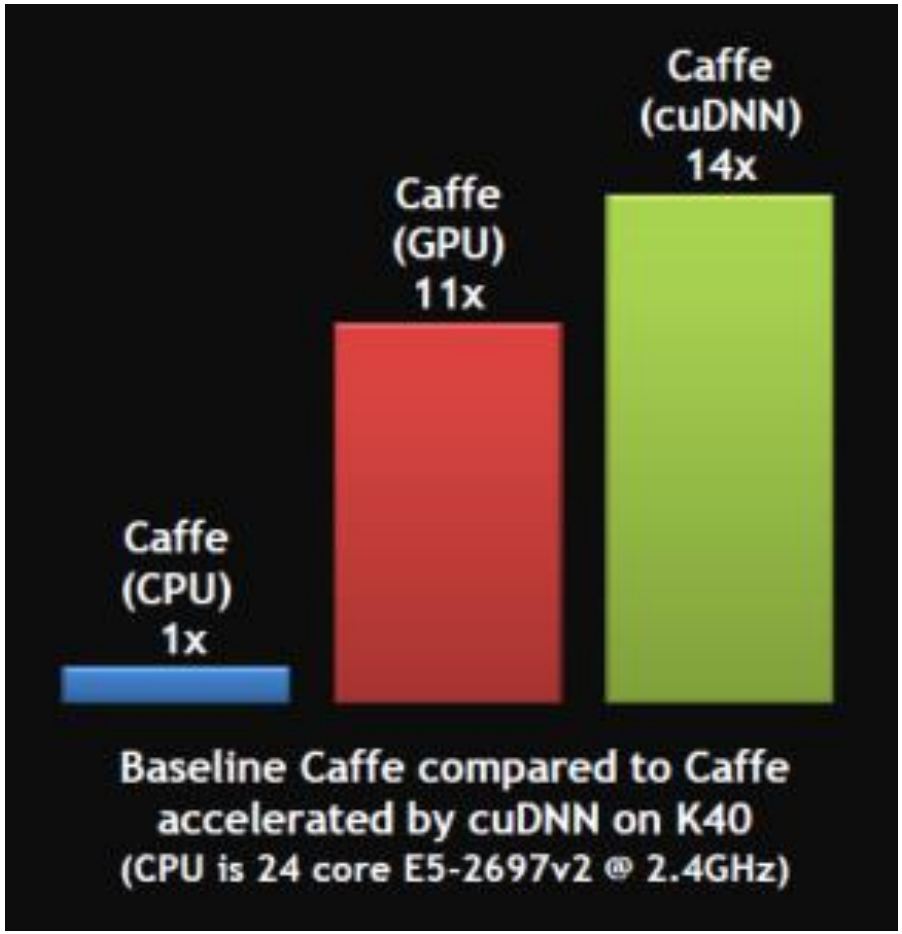
Подвыборка – операция уменьшения размерности. Чаще всего – выбор максимума или поиск среднего по нескольким соседним нейронам.

Свертка

Подвыборка

Свертка

cuDNN - нейронные сети глубокого обучения



Другие библиотеки

NVBIO

FFmpeg

OpenCV

MAGMA

ICL UT

API: C++
FORTRAN

Plug-ins:
MATLAB/Octave, Deal.II,
OpenFOAM, Hermes, Elmer, ...

PARALUTION

Multi-core CPU
OpenMP

GPU
CUDA/OpenCL

Xeon Phi
OpenMP

[HiPLAR]