Open in Colab

# Thrust Library

According to nVidia:

Thrust is a C++ template library for CUDA based on the Standard Template Library (STL). Thrust allows you to implement high performance parallel applications with minimal programming effort through a high-level interface that is fully interoperable with CUDA C.

Thrust provides a rich collection of data parallel primitives such as scan, sort, and reduce, which can be composed together to implement complex algorithms with concise, readable source code. By describing your computation in terms of these high-level abstractions you provide Thrust with the freedom to select the most efficient implementation automatically. As a result, Thrust can be utilized in rapid prototyping of CUDA applications, where programmer productivity matters most, as well as in production, where robustness and absolute performance are crucial.

## Reduction

In [1]:
```
%%file thrust_sum.cu

#include <stdio.h>
#include <thrust/device_vector.h>

void cpu_sum(int *x, int n)
{
    int result = 0;
    for(unsigned int i=0; i < n; ++i) {
        result += x[i];
    }
    printf("CPU Sum is %d \n", result);
}

void gpu_sum(int *x, int n)
{

    thrust::device_vector<int> d_vec(n,0); // initialize all n integers of a device_vector to 0

    for(unsigned int i = 0; i < n; ++i){
        d_vec[i] = x[i];
    }

    int t_sum = thrust::reduce(d_vec.begin(), d_vec.end(), (int) 0, thrust::plus<int>());
    printf("GPU (thrust) Sum is %d \n", t_sum);
}


int main()
{
    int h[] = {10, 1, 8, -1, 0, -2, 3, 5, -2, -3, 2, 7, 0, 11, 0, 2};

    int size = sizeof(h);
    int count = size/sizeof(int);

    cpu_sum(h, count);
    gpu_sum(h, count);

    return 0;
}
```

```
Writing thrust_sum.cu
```

In [2]:
```
!nvidia-smi
```

```
Wed Feb 23 19:30:25 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P8    26W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

In [4]:
```
%%bash

CUDA_SUFF=35
nvcc -gencode arch=compute_${CUDA_SUFF},code=sm_${CUDA_SUFF} ./thrust_sum.cu -o thrust_sum
./thrust_sum
```

```
CPU Sum is 41
GPU (thrust) Sum is 41
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are dep
recated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppress warning).
```

## SAXPY

**SAXPY** stands for “Single-Precision A·X Plus Y”. It is a function in the standard Basic Linear Algebra Subroutines (BLAS)library.

In [5]:
```
%%file thrust_saxpy.cu

#include <stdio.h>
#include <thrust/device_vector.h>
#include <iostream>

struct saxpy_functor
{
    const float a;
    saxpy_functor(float _a) : a(_a) {}

    __host__ __device__
    float operator()(const float& x, const float& y) const
    {
        return a * x + y;
    }
};

thrust::device_vector<float> saxpy_fast(float A, thrust::device_vector<float>& X, thrust::device_vector<float>&
{
    // SAXPY stands for “Single-Precision A·X Plus Y”
    // result <- A * X + Y
    thrust::device_vector<float> result(X.size());
    thrust::transform(X.begin(), X.end(), Y.begin(), result.begin(), saxpy_functor(A));

    return result;
}

int main(void)
{
    // allocate two device_vectors with 5 elements
    thrust::device_vector<float> X(5);
    thrust::device_vector<float> Y(5);

    // initialize the arrays to 0,1,2,3,4
    thrust::sequence(X.begin(), X.end());
    thrust::sequence(Y.begin(), Y.end());

    auto result = saxpy_fast(100, X, Y);

    // print results
    for(int i = 0; i < result.size(); i++)
        std::cout << "result[" << i << "] = " << result[i] << std::endl;
}

// output
// result[0] = 0
// result[1] = 101
// result[2] = 202
// result[3] = 303
// result[4] = 404
```

```
Writing thrust_saxpy.cu
```

In [6]:
```
%%bash

CUDA_SUFF=35
nvcc -gencode arch=compute_${CUDA_SUFF},code=sm_${CUDA_SUFF} ./thrust_saxpy.cu -o thrust_saxpy
./thrust_saxpy
```

```
result[0] = 0
result[1] = 101
result[2] = 202
result[3] = 303
result[4] = 404
nvcc warning : The 'compute_35', 'compute_37', 'compute_50', 'sm_35', 'sm_37' and 'sm_50' architectures are dep
recated, and may be removed in a future release (Use -Wno-deprecated-gpu-targets to suppress warning).
```

## Additional reading

Official Quick Start Guide.