Thrust

http://docs.nvidia.com/cuda/thrust/

- OpenSource Library
- Highly optimized algorithms
- Header only (facilitates portability, but needs recompiling)

Documentation page !

- http://thrust.github.io/doc/

Several interesting components

- Device container : thrust::device_vector
  - ✴ Associated with device_ptr and raw_pointer (see TP)
- Useful algorithms
  - ✴ Transformations
  - ✴ Reductions
  - ✴ Sort
- Random number generation
- Unsynchronized

How to use

- Already in the distribution
- #include <thrust/…>
- optional :
  - ✴ using namespace thrust;

Cases

Folder THRUST

We use thrust::device_vector<T> (T = int, float…)

Case 0 : device_pointers

- Main containers of thrust
- Can be created in many ways
- Demonstration in THRUST/device_ptrs.cu

Case 1 : simple transformation

- Generate two sequences of number, S1 and S2, starting form 1 to 20 with step 1
- Transform each element by negating it
- Print the resulting array using (to adapt !) printme<<<1,1>>>(<pointer to device array>, <pointer to device array>);

Case 2 : simple scalar product

- Create a sequence (thrust::device_vector<>) of number v1 starting form 1 to N with step 2, and another v2 with only 1’s. Use thrust::sequence and thrust::fill

- Compute their scalar product

  ✳ Hint : check the function inner_product

- Try also using algorithms sequence and fill using a thrust::device_ptr<>

Case 3 : random number generation

- Goal : generate two random gaussian vector of size N (suggestion : N =10^6) with mean 0 and variance 1 and 2, i.e N(0,1) and N(0, 2).
- Calculate their mean, variance.
- Consider first vector.
- Remove the values whose absolute value is above a threshold.
  - Hint : combine thrust::erase and thrust::remove_if
  - Count remaining elements.
- Sort the data.
- Copy back the result to the CPU.

Be careful for performance even with libraries !

http://docs.nvidia.com/cuda/thrust/#axzz3ar2opcwL

- Saxpy example

Additional examples

- https://github.com/thrust/thrust/tree/master/examples
- Included in THRUST folder

Fancy iterators

- http://docs.thrust.googlecode.com/hg/group__fancyiterator.html