MI3.22 – Advanced Programming for HPC
Labwork 3 – *Parallel Patterns with Thrust*

Exercise 1: List selection

Let L be an array that contains n colored objects. Each object is blue or red. The question is to build an efficient parallel algorithm that returns a new array with only blue objects (so, a COMPACTION).

In this exercise, it is requested to not use the function `thrust::copy_if`, but only `SCAN`, `SCATTER_IF`, and basic `MAP` (aka transform).

Exercise 2: Radix Sort

The radix sort is a very efficient sorting algorithm on parallel computers. Indeed, the number of outer loops is known in advance, and corresponds to the number of bits of the maximum value in the array to sort. Generally speaking, with 32 bits values, radix sort needs 32 outer loops.

1. In order to understand this sorting algorithm, we starts with a simple tutorial exercise.

The array $A = [a_0, a_1, \ldots, a_{n-1}]$ contains n integers. We define two following operators:

- Prescan(A), that returns the array $[0, a_0, a_0 + a_1, \ldots, a_0 + a_1 + \ldots + a_{n-2}]$;
- Scan(A), that returns the array $[a_0, a_0 + a_1, \ldots, a_0 + a_1 + \ldots + a_{n-1}]$.

Section "Pointers jumps" of lecture 2 has shown their implementations in $O(\log n)$ with a P-RAM EREW.

1. Using a boolean array `Flags`, what does the following procedure?

```
SPLIT($A$, Flags)
    Idown ← PRESCAN($not$(Flags))
    Iup ← $n -$ REVERSE(SCAN(REVERSE(Flags)))
    For $i = 1$ To $n$ in parallel:
        If Flags($i$) Then
            Index[$i$] ← Iup[$i$]
        Else
            Index[$i$] ← Idown[$i$]
    Return PERMUTE($A$, Index)
```

The used functions are relatively intuitive. An example of used is:

$$
\begin{array}{rcl}
A & = & [\; 5 \; 7 \; 3 \; 1 \; 4 \; 2 \; 7 \; 2 \;] \\
\texttt{Flags} & = & [\; 1 \; 1 \; 1 \; 1 \; 0 \; 0 \; 1 \; 0 \;] \\
\texttt{Idown} & = & [\; 0 \; 0 \; 0 \; 0 \; 0 \; 1 \; 2 \; 2 \;] \\
\texttt{Iup} & = & [\; 3 \; 4 \; 5 \; 6 \; 7 \; 7 \; 7 \; 8 \;] \\
\texttt{Index} & = & [\; 3 \; 4 \; 5 \; 6 \; 0 \; 1 \; 7 \; 2 \;] \\
\text{Result} & = & [\; 4 \; 2 \; 2 \; 5 \; 7 \; 3 \; 1 \; 7 \;]
\end{array}
$$

What is the cost of the Split function?

2. Let now the following Mystery procedure:

```
MYSTERY($A$, $NumberOfBits$)
    For $i = 0$ To $NumberOfBits - 1$:
        bit($i$) ← array indicating if the i$^{th}$ bit of the elements of $A$ is set to 1
        $A$ ← SPLIT($A$, bit($i$))
```

(a) Test this mysterious procedure with the array $A = [5, 7, 3, 1, 4, 2, 7, 2]$ using only 3 bits.

(b) What does this procedure Mystery ?

(c) Using entry of size $O(\log n)$ bits, what is the complexity using n processors? And with only p processors? Which are the most interesting values of p?

2. Implements this algorithm using Thrust. Notice that the `thrust::sort` method does exactly that with a very efficient implementation. Nevertheless, again you can only use `SCAN`, `GATHER` or `SCATTER`, and basic `MAP` (aka transform).

Exercise 3: Parallel Quicksort

Write a parallel quick-sort in EREW PRAM. Then, write the implementation using Thrust.