More Algorithms

This section introduces two more algorithms, the fused and scan algorithms!

WE'LL COVER THE FOLLOWING

- `transform_reduce` - Fused Algorithm
- Scan Algorithms

`transform_reduce` - Fused Algorithm

To get even more flexibility and performance, the `reduce` algorithm also has a version where you can apply a transform operation before performing the reduction.

```
int main() {
  std::vector<int> v{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
  auto sumTransformed = std::transform_reduce(std::execution::par,
      v.begin(),
      v.end(),
      0,
      std::plus<int>{},
      [](const int& i) { return i * 2; }
  );
  cout << sumTransformed;
  return 0;
}
```

The above code will first execute the unary functor - the lambda that doubles the input value; then the results will be reduced into a single sum.

The fused version will be faster than using two algorithms: `std::reduce` firstly and then `std::reduce` - because the implementation will need to perform the parallel execution setup only once.

Scan Algorithms

The third group of new algorithms is `scan`. They implement a version of partial sum, but out of order.

The exclusive scan does not include the i-th element in the output i-th sum, while inclusive scan does.

For example for `array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}`, we'll get the following values for partials sums:

Index	0	1	2	3	4	5	6	7	8	9
Values	1	2	3	4	5	6	7	8	9	10
Exclusive partial sums	0	1	3	6	10	15	21	28	36	45
Inclusive partial sums	1	3	6	10	15	21	28	36	45	55

Similarly to `std::reduce`, the order of the operations is unsequenced, so to get deterministic results the `binary_op` must be associative.

`scan` has also two fused algorithms: `transform_exclusive_scan` and `transform_inclusive_scan`. Both of the algorithms will perform a unary operation on the input container, and then they will compute the prefix sums on the output.

Prefix sums have an essential role in many applications, for example for stream compaction, computing summed area tables or radix sort. Here's a link to an article that describes the algorithms in detail.

Now that we have studied parallel algorithms in detail, let's look at their performance in the next lesson.