

Parallel STL

You can choose the execution policy of an algorithm.

- Execution policies

- `std::execution::seq`

- Sequential in one thread

- `std::execution::par`

- Parallel

- `std::execution::par_unseq`

- Parallel and vectorized  SIMD

Parallel STL

```
const int SIZE = 8;
int vec[]={1, 2 , 3, 4, 5, 6, 7, 8};
int res[SIZE] = {0,};

int main(){
    for (int i= 0; i < SIZE; ++i){
        res[i] = vec[i] + 5;
    }
}
```

Not vectorized

```
movslq  -8(%rbp), %rax
movl     vec(,%rax,4), %ecx
addl     $5, %ecx
movslq  -8(%rbp), %rax
movl     %ecx, res(,%rax,4)
```

Vectorized

```
movdqa   .LCPI0_0(%rip), %xmm0    # xmm0 = [5,5,5,5]
movdqa   vec(%rip), %xmm1
paddq    %xmm0, %xmm1
movdqa   %xmm1, res(%rip)
paddq    vec+16(%rip), %xmm0
movdqa   %xmm0, res+16(%rip)
xorl     %eax, %eax
```

Parallel STL

```
std::vector<int> vec = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

```
std::sort(vec.begin(), vec.end());    // sequential as ever
```

```
sort(std::execution::seq, vec.begin(), vec.end()); // sequential
```

```
sort(std::execution::par, vec.begin(), vec.end()); // parallel
```

```
sort(std::execution::par_unseq, vec.begin(), vec.end()); // par + vec
```

Parallel STL

adjacent_difference, adjacent_find, all_of, any_of, copy, copy_if, copy_n,
count, count_if, equal, **exclusive_scan**, fill, fill_n, find, find_end,
find_first_of, find_if, find_if_not, **for_each**, **for_each_n**, generate,
generate_n, includes, **inclusive_scan**, inner_product, inplace_merge,
is_heap, is_heap_until, is_partitioned, is_sorted, is_sorted_until,
lexicographical_compare, max_element, merge, min_element, minmax_element,
mismatch, move, none_of, nth_element, partial_sort, partial_sort_copy,
partition, partition_copy, **reduce**, remove, remove_copy, remove_copy_if,
remove_if, replace, replace_copy, replace_copy_if, replace_if, reverse,
reverse_copy, rotate, rotate_copy, search, search_n, set_difference,
set_intersection, set_symmetric_difference, set_union, sort,
stable_partition, stable_sort, swap_ranges, transform,
transform_exclusive_scan, **transform_inclusive_scan**, **transform_reduce**,
uninitialized_copy, uninitialized_copy_n, uninitialized_fill,
uninitialized_fill_n, unique, unique_copy