### Algorithm Update

This section lists all the algorithms that can handle execution policy parameter. It further looks at the usage of the Run and Measure function.

#### WE'LL COVER THE FOLLOWING ^

- Execution Policy Parameter
- New Algorithms

## **Execution Policy Parameter** #

The execution policy parameter was added to most of the existing algorithms.

#### Here's the list:

adjacent_difference	inplace_merge	replace_copy
adjacent_find	is_heap	replace_copy_if
all_of	is_heap_until	replace_if
any_of	is_partitioned	reverse
сору	is_sorted	reverse_copy
copy_if	is_sorted	rotate
copy_n	is_sorted_until	rotate_copy
count	lexicographical_comp	search

count_if	max_element	search_n
equal	merge	set_difference
exclusive_scan	min_element	set_intersection
fill	minmax_element	set_symmetric_differ ence
fill_n	mismatch	set_union
find	move	sort
find_end	none_of	stable_partition
<pre>find_first_of</pre>	nth_element	stable_sort
<pre>find_if</pre>	partial_sort	swap_ranges
<pre>find_if_not</pre>	partial_sort_copy	transform
for_each	partition	transform_exclusive_
for_each_n	partition_copy	transform_inclusive_
generate	remove	transform_reduce
generate_n	remove_copy	uninitialized_copy
includes	remove_copy_if	uninitialized_copy_n
<pre>inclusive_scan</pre>	remove_if	uninitialized_fill

<pre>inner_product</pre>	replace	uninitialized_fill_n	
	unique	unique_copy	

# New Algorithms #

To fully support new parallel execution patterns The Standard Library was also equipped with a set of new algorithms:

Algorithm	Description
for_each	similar to for_each except returns void
for_each_n	applies a function object to the first  n elements of a sequence
reduce	similar to accumulate, except out of order execution to allow parallelism
transform_reduce	transforms the input elements using a unary operation, then reduces the output out of order
exclusive_scan	parallel version of <a href="mailto:partial_sum">partial_sum</a> , excludes the i-th input element from the i-th sum, out of order execution to allow parallelism
inclusive_scan	parallel version of <a href="mailto:partial_sum">partial_sum</a> , includes the i-th input element in the i-th sum, out of order execution to allow parallelism
transform exclusive scan	applies a functor, then calculates

cransion m_cxctastvc_scan	exclusive scan
transform_inclusive_scan	applies a functor, then calculates inclusive scan

The new algorithms form three groups: for\_each, reduce and then scan, plus their alternatives.

With reduce and scan you also get "fused" versions like transform\_reduce. These compositions should give you much better performance than using two separate steps - because the cost of parallel execution setup is smaller and also you have less one loop traversals.

The new algorithms also provide overloads without the execution policy parameter so that you can use them in a standard serial version.

In the next lesson, you'll find a description of each group.