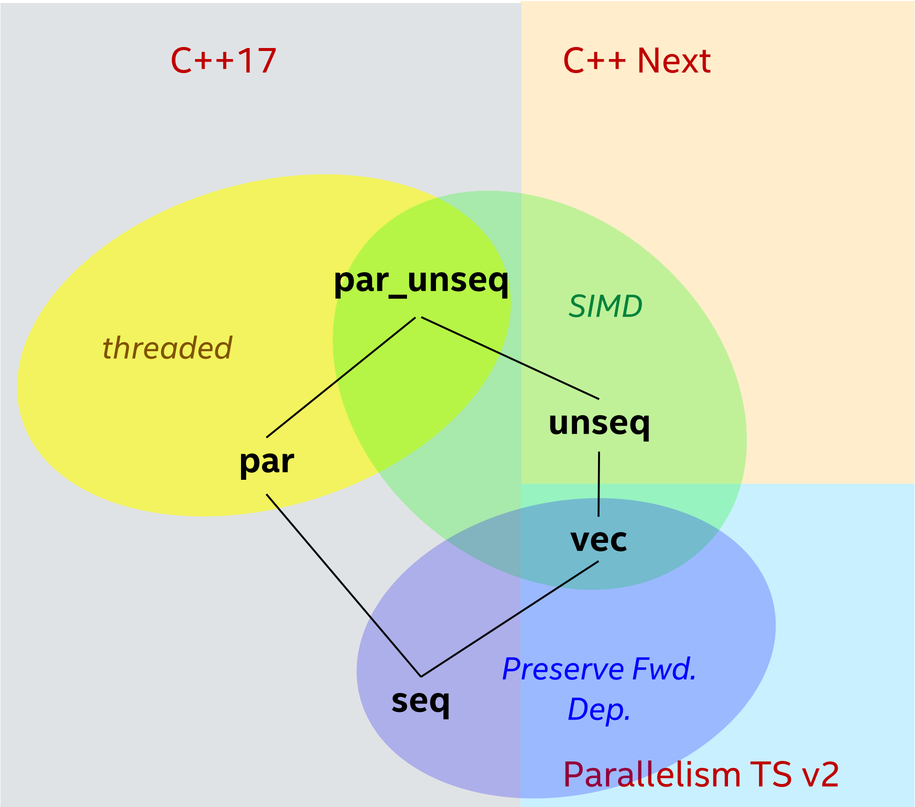
Software

PARALLEL STL IN 5 MINUTES

Mikhail Dvorskiy, Jim Cownie, Alexey Kukanov

What is the Parallel STL?

- An extension of the C++ Standard Template Library algorithms with the “execution policy” argument
- Support for parallel execution policies is in the C++17 standard
- Support for the unseq policy is on track for the next C++ standard
- Support for vector policies is being developed in the Parallelism Technical Specification (TS) v2 which has just been approved for an ISO vote



Example of Use

Additions to existing STL code

```
#include <algorithm>
#include <execution>
……
//Using outer parallelization and inner vectorization
void Image::ApplyGamma( float g ) {
    using namespace std;
    using namespace std::execution;
    for_each( par, image.begin(), image.end(), [g]( Row &r ) {
        transform( unseq, r.cbegin(), r.cend(), r.begin(),
            [g]( float v ) { return pow( v, g ); } ); //should be vectorizable
    } );
}
```

- The version of “transform” is resolved by ExecutionPolicy type (see [algorithms.parallel.overloads]):

```
template <class ExecutionPolicy, class InputIt, class OutputIt, class UnaryOp>
OutputIt transform(ExecutionPolicy&& exec, …);
```

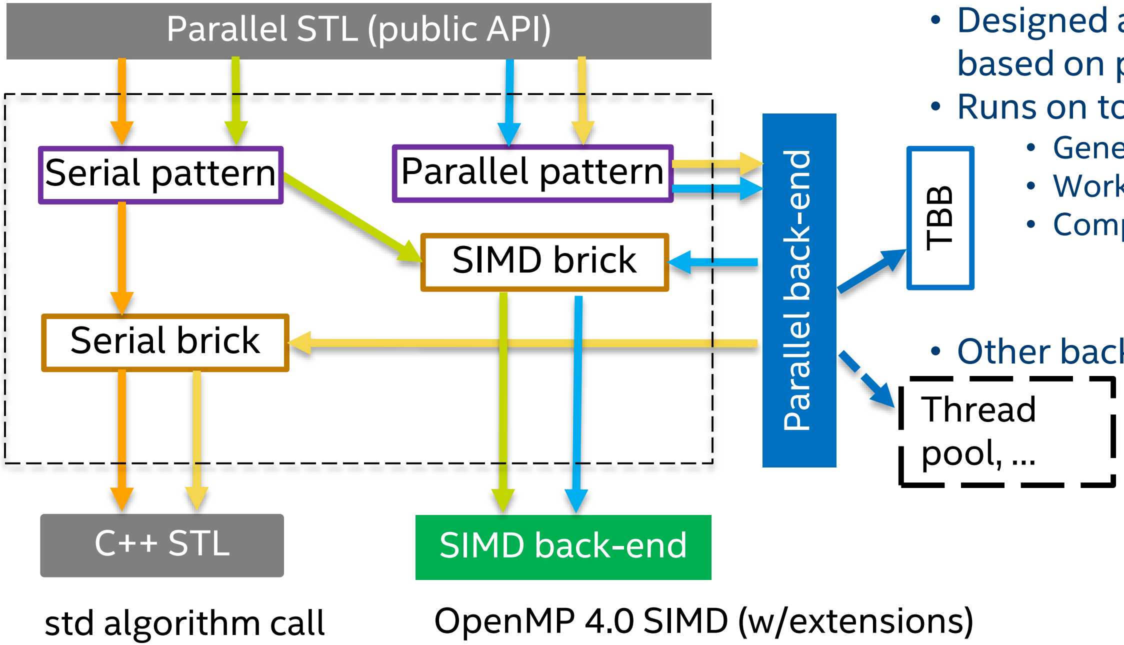
- Beware: for vector policies, the lambda/functor/predicate should be vectorizable

Parallel STL advantages

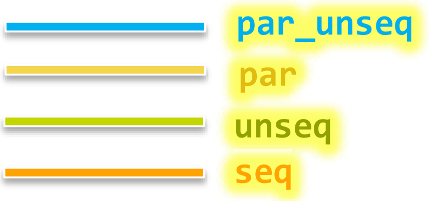
Compared with other parallelization tools and libraries

- Standard parallelization tool coming with C++ 17
- Simple high-level API
- Can express two different styles of parallelism (with more to come)
- Compile-time dispatch: no runtime overhead
- Facilitates correctness (no races if used as expected)
- Scalability & Composability
  - depends on implementation back-end

Intel's Parallel STL implementation



- Four policies lead to four different implementations of each algorithm
- Designed as combination of "pattern & brick" pairs based on policy traits (`is_parallel`, `is_vector`)
- Runs on top of Threading Building Blocks (TBB):
  - Generic parallel patterns in C++
  - Work balancing scheduler
  - Composable and portable
- Other backends can be implemented



How to Get Involved

- Parallel STL main repository (upstream) at GitHub: https://github.com/intel/parallelstl
  - You can contribute by sending patches or preparing pull requests
- Intel contributes the implementation to both GCC and LLVM
  - GCC community is adjusting the code for use in libstdc++ (WIP)
  - **LLVM community can adjust the code for use in libc++**
    - Integrate codebases, adjust/extend/add tests, etc.
  - Communities contribute integration changes upstream
- Contact us (via inteltbbdevelopers@intel.com) if you are interested but don’t know where to start
- We want your help! (LLVM is behind GCC here ☹)

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

intel®

Software