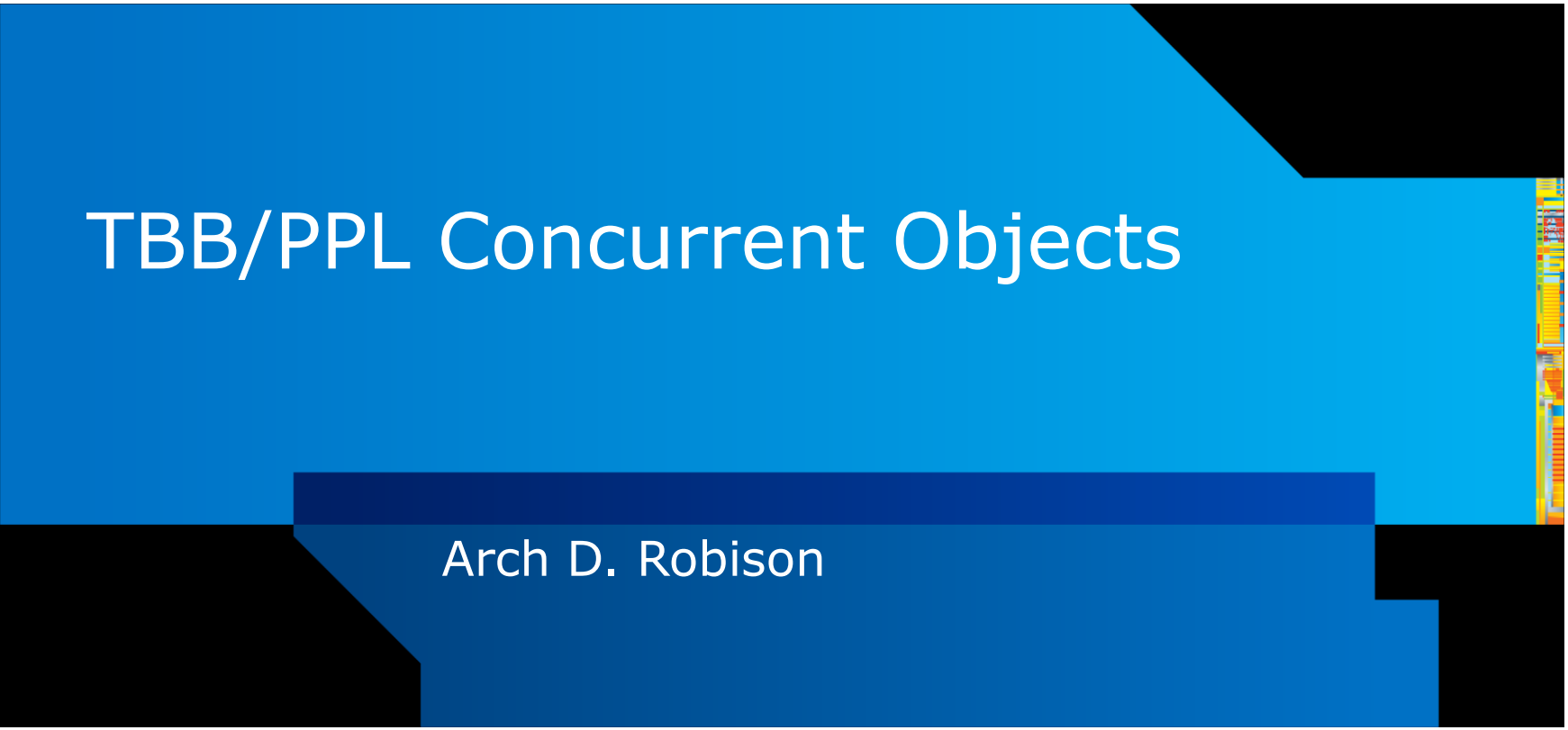
TBB/PPL Concurrent Objects
Arch D. Robison

Concurrent Objects

Support for concurrent sharing and updating

Higher concurrency than STL container protected by a mutex

- Non-blocking or fine-grain locking
- Literature has numerous implementation techniques

Preserve key invariants of the container

- Interfaces resemble sequential STL, but necessarily depart
- Does not preserve invariants between items or between containers
  - Not a cure-all for lock-free programming, but nonetheless useful.

Example: concurrent_queue

Operations must be serialized.

```
extern std::queue<T> q;
if(!q.empty()) {
    item=q.front();
    q.pop();
}
```

Concurrent pushes/pops allowed.

```
extern concurrent_queue<T> q;
q.try_pop(item);
```

Idiomatic sequence packaged as single linearizable method.

Invariant:

- If q.push(a) happens before q.push(b), then q.pop(b) cannot happen before q.pop(a).

Containers

Object	Concurrent Ops
concurrent_queue concurrent_priority_queue	push, pop
concurrent_unordered_set concurrent_unordered_map concurrent_unordered_set concurrent_unordered_multimap	insert, find, iterate*
concurrent_vector	push_back, operator[], grow_by, grow_to_at_least

*concurrent erase possible in theory, but practically useless without garbage collector.

Example: concurrent_vector

Serial append contents of u to v

```
extern std::vector<T> v;
v.insert( v.end(), u.begin(), u.end() );
```

Concurrent append contents of u to v.

```
extern concurrent_vector<T> v;
copy( u.begin(), u.end(), v.grow_by(u.size()) );
```

Implementation note: Elements are not contiguous in memory.

Combinable

Useful for parallel reductions

- Each thread gets its own thread-local value
  - Value initialized on first touch
- Method **`combine`** combines the values
  - Reduction operation should be associative *and* commutative

Summing f(i)

```
combinable<float> sum;
parallel_for( 0, n, [&](int i){
    sum.local() += f(i);
});
y = sum.combine( std::plus<float>() );
```

Open Issues

Value-oriented associative containers?

- STL has reference-based associative containers.
- Value-oriented associative containers would mostly solve concurrent-erasure problem.

Separate “parallel” and “serial” views?

- Serial view might permit faster operations.

Selecting concurrent requirements more precisely

- E.g. single-producer single-consumer queue

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.

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