Implementations of concurrent data structures consist of multiple interdependent functions; as a result, the complexity of common operations can be increased due to the overhead needed to provide for the correctness of another, perhaps unrelated operation. However, this overhead can be reduced in applications which do not need the full set of operations and concurrency guarantees supported by a typical parallel container. To take advantage of such scenarios, several versions of a data container with reduced functionality can be combined into a library where, by using program analysis, the appropriate data structure is chosen to fit the use case.

**Current Practice**

Software engineers do not have access to a library of disjoint-parallel containers for C++.
- The C++ Standard Template Library does not provide thread-safe containers.
- No one-size-fits-all solution is flexible enough to perform well in all use cases.
- Most projects requiring high performance have to devote time to creating data structures in the beginning.

**Related Work**

A brief summary is provided.
- Intel’s TBB provides fine-grained locking containers, but its runtime system can degrade performance.
- Boost provides limited support for thread-safe containers.
- The Standard Template Adaptive Parallel Library is designed to adapt to specific use cases, but it adds this overhead in the form of a runtime system which can have a severe performance impact.
- SPIN provides the ability to verify arbitrary correctness conditions, but requires users to create a model of their code in Promela.

**Overview**

- Use concurrent data structures, e.g., vector.
- Analyzes source codes and chooses an implementation from a repository that is a best fit for a scenario and performance model.
- User may annotate code with information that cannot be automatically deduced.
- Programmer contains families of data structures (e.g., vector) and implementations that are optimized for specific use cases.
- Source Code Analysis and Rewrite System
- Repository
- Source files
- Input: original code
- Output: optimized code
- Full Vector
- Random/Access + push Vector
- Full Vector w/o erase
- Use best fitting instance of a concurrent data structure. E.g., full vector without erase.

**Tervel**

We have developed a framework that provides non-blocking memory management and progress assurance, which allows users to develop lock-free and wait-free algorithms.
- This framework is open source and can be found at: [http://ucf-cs.github.io/Tervel/](http://ucf-cs.github.io/Tervel/)
- Tervel comes bundled with implementations of our existing non-blocking data structures, such as: a hash map, a stack, two ring buffers, a vector, and a multi-word compare-and-swap.

**CCSpec**

We have developed a tool for verifying correctness conditions that can be specified in terms of method orderings.
- This tool works on annotated source files that use the C/C++11 atomics and threading library.
- The user needs to specify the beginning, end, input, and output of the function, in addition to the correctness condition itself.
- We verified the following conditions of data structures with our tool: linearizability, sequential consistency, quiescent consistency, and quasi-linearizability.
- This tool is open source: [http://ucf-cs.github.io/CCSpec/](http://ucf-cs.github.io/CCSpec/)

**Community**

We will foster a community of data structure users and developers.
- Software engineers will able to learn about our project through a massive open online course (MOOC).
- A website with documentation and a forum will be created and regularly maintained.
- This work will be presented at conferences, workshops, seminars, and industry meetings.

**Impact**

This research will use and advance multiprocessor algorithms development, program analysis, and formal specification and reasoning to provide a pragmatic and easily accessible software design capability.

The proposed methodology will benefit commercial and scientific software design, thus having an immediate impact on the advancement of science and economic growth.