LC/DC: Lockless Containers and Data Concurrency

Damian Dechev

cse.eecs.ucf.edu/LCDC

Computer Software Engineering
Scalable & Secure Systems Lab

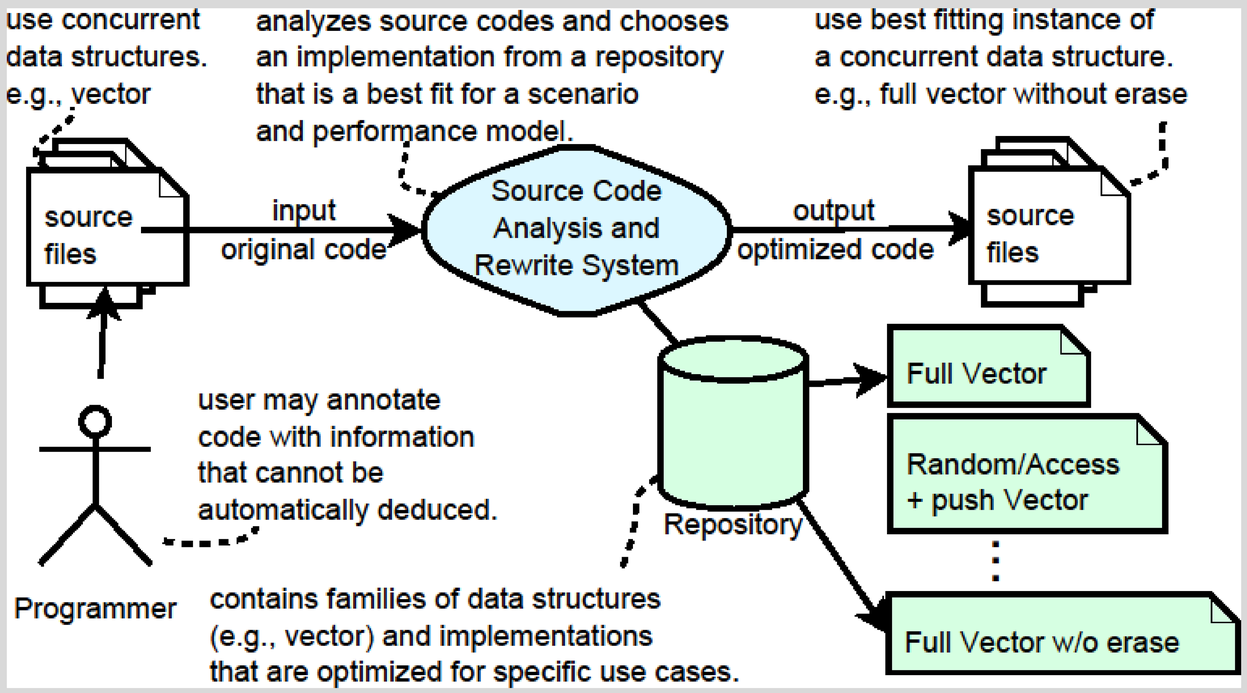
Implementations of concurrent data structures consist of multiple interdependent functions; as a result, the complexity of common operations can be increased due to the overhead needed to provide for the correctness of another, perhaps unrelated operation. However, this overhead can be reduced in applications 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

Overview



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

Related Work

A brief summary is provided.

* Intel's TBB provides fine-grained locking containers, but it's 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

Specification Language

We will develop a specification language that allows the user to describe the properties of their data structure in an intuitive way with an underlying formal logic.

* This specification language will be designed to be easily translatable to a modeling language, so that the user's specification can be formally verified
* Design challenges include defining axioms that allow the user to specify lock-freedom and linearizability in an easy to use way

Profiling Tools

We will develop a set of profiling tools that will allow us to choose the best data structure from a family of reduced functionality containers.

* These tools will aid our selection of the correct data structure by accounting for performance differences caused by use case and hardware details
* An example of a relevant statistic is the number of cache misses, because these could be symptomatic of false sharing

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.