Ristra Atomics Library

Background. The Standard C++ Library's <atomic> header defines an atomic<T> class template with basic functionality for any trivially-copyable type T; informally, any type T for which a shallow copy, as with memcpy, has the correct semantics. Integral and pointer (but not floating-point) specializations support &=, |=, and ^= (integral only); += and -=; and prefix and postfix ++ and --. Other viable operations, such as *= and <<=, are not provided.

Sandia National Laboratory's *Kokkos* provides a variety of atomic operations in functional form. Kokkos::atomic_fetch_add(x,y), for instance, is an atomic x += y. As with C++, some viable operations are missing. Floating-point data are supported, but pointers are not.

Ristra Atomics. Our new, "header-only" Ristra Atomics C++ library provides a class template ristra::atomics::atomic<T,SCHEME>, a superset of the atomic operations of C++ and Kokkos, and several available "atomicity schemes" for achieving the requisite atomicity. Schemes include our own general ones, as well as forwards to C++ or Kokkos capabilities (Kokkos inclusion is optional) when available. A scheme is specified by providing any of several available tag classes as atomic's second template argument. This compile-time mechanism facilitates function overloading and inlining, giving runtime efficiency.

Atomicity Schemes. Ristra Atomics provides the following atomicity schemes.

cpp
Forward to the C++ library's capability.

kokkos
Forward to the Kokkos library's capability.

lock
Embed the operation in a lock_guard<mutex>.

strong
Atomicize the operation via a strong compare-and-swap (CAS) loop.

strong::pun
Like strong, but with T punned to a same-sized integral type for CAS. May give better performance on some architectures.

weak
Like strong, but using a weak CAS.

weak::pun
Like strong::pun, but using a weak CAS.

serial
Perform the operation without atomicizing it.

Operations. Atomic operations in Ristra Atomics are available in several forms. Operators: +=, -=, *=, /=, %=, <<=, >>=, &=, |=, ^=, ++value, value++, --value, and value--. Functions and member functions: add, sub, mul, div, mod, lshift, rshift, andeq, oreq, xoreq, preinc, postinc, predec, and postdec. **Example**: for x of atomic type, x.add(2) and add(x,2) both perform x += 2 atomically. Also available: inc == preinc, dec == predec, min, and max. **Example**: x.min(y) computes x = min(x,y) atomically.

Data Types. Integral, floating-point, pointer, and general trivially-copyable data are all supported. Bear in mind that the availability of a specific atomic computation necessarily depends on the data type, atomicity scheme, and operation. The kokkos scheme doesn't support pointers, while cpp doesn't support floating-point values. No scheme supports bitwise operations on floating-point values, which in C++ have no bitwise operations to begin with. Only our own schemes support operations on values of user-defined types – and, then, only if the type has been outfitted with a non-atomic version of the operation.

Summary. Ristra Atomics is an easy-to-use, header-only C++ library for performing atomic operations on integral, pointer, floating-point, and general trivially-copyable data. It supports several of its own atomicizing schemes, and provides access to the capabilities in C++ and in Kokkos. The library thus brings together two existing sets of capabilities, adds more, and embeds them all into a coherent notational and logical interface.