STUDY OF IMPLEMENTATION OF NONBLOCKING K-COMPARE-SINGLE-SWAP IN JAVA

There exist two extremes of nonblocking software synchronization support for concurrent data structure design: intricate designs of specific structures based on single-location operations such as **compare-and-swap (CAS)**, and general-purpose multilocation **transactional memory** implementations. While the former are sometimes efficient, they are invariably hard to extend and generalize. The latter are flexible and general, but costly. This paper aims at a middle ground: reasonably efficient multilocation operations that are general enough to reduce the design difficulties of algorithms based on CAS alone.

The authors of this paper present an obstruction-free implementation of an **atomic k-location-compare single-swap (KCSS)** operation. KCSS allows for simple nonblocking manipulation of linked data structures by overcoming the key algorithmic difficulty in their design: making sure that while a pointer is being manipulated, neighboring parts of the data structure remain unchanged. Our algorithm is efficient in the common uncontended case: A successful k-location KCSS operation requires only **two CAS** operations, **two stores**, and **2k noncached loads** when there is no contention.

REFERENCES

1. V. Luchangco, M. Moir, and N. Shavit. Nonblocking k-compare-single-swap. SPAA'03, June 7–9, 2003.

2. Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit. Atomic snapshots of shared memory. Journal of the ACM (JACM), 40(4):873–890, 1993.