Tracking in Order to Recover — Detectable Recovery of Lock-Free Data Structures*

Hagit Attiya†
Technion
hagit@cs.technion.ac.il

Ohad Ben-Baruch†
Ben-Gurion University
ohadben@post.bgu.ac.il

Panagiota Fatourou
FORTH ICS &
University of Crete, CSD
faturu@csd.uoc.gr

Danny Hendler†
Ben-Gurion University
hendlerd@cs.bgu.ac.il

Eleftherios Kosmas‡
University of Crete, CSD
ekosmas@csd.uoc.gr

ABSTRACT

This paper presents the *tracking approach* for deriving *detectably recoverable* (and thus also *durable*) implementations of many widely-used concurrent data structures. *Info-Structure Based* (*ISB*)-tracking amends descriptor objects used in existing lock-free helping schemes with additional fields that track an operation's progress towards completion and persists these fields to memory in order to ensure detectable recovery. We have applied ISB-tracking to derive detectably recoverable implementations of a queue, a linked list, a binary search tree, and an exchanger. Experimental results show the feasibility of the technique.

1 MOTIVATION

Byte-addressable *non-volatile main memory* (*NVRAM*) combines the performance benefits of conventional main memory with the durability of secondary storage. Systems with NVRAM will be more prevalent in the near future. The availability of durable main memory has increased the interest in the *crash-recovery* model, in which failed processes may be resurrected after the system crashes. Of particular interest is the design of *recoverable concurrent data structures*, whose operations can recover from crash-failures. Such data structures are important as they are building blocks for constructing simple, well-structured, sound and error-resistant multiprocessor systems. For example, in many big-data applications, shared in-memory tree-based data indices are created for fast data retrieval and useful data analytics.

When designing recoverable data structures, it is important to be able to tell after recovery whether an operation was executed to completion and if so, what its response was, a property called *detectable recovery* [1, 2]. In many computer systems (e.g., databases), detectable recovery is supported by precisely *logging* the progress of computations to non-volatile storage, and replaying the log during recovery. Logging imposes significant overheads in time and space. This cost is even more pronounced for concurrent data structures, where there is an extra cost of synchronizing log accesses.

2 CONTRIBUTION AND METHODOLOGY

We present the *Info Structure Based (ISB) tracking approach* for deriving *detectable* implementations of many widely-used concurrent data structures for systems with *non-volatile main memory* (*NVRAM*). ISB tracking avoids full-fledged logging, and tracks the progress of each operation individually, in a way supporting detectable recovery. Specifically, it explicitly maintains an *Info structure*, stored in non-volatile memory, to track an operation's progress as it executes. The Info structure allows a process to decide, upon recovery, whether the operation's effect has already become visible to other processes, in which case, the mechanism allows to determine the response of the operation. ISB-tracking is widely applicable—it can be used to derive recoverable versions of a large collection of concurrent data structures, including concurrent tree-like structures that could be used as indices.

We emphasize that detectability is a challenge even if caches are non-volatile, i.e., writes are immediately persisted, in program order. However, ISB-tracking informs how *persistency instructions* (flushes and fences) should be inserted for ensuring an implementation's correctness in an efficient manner, even when cache memories are volatile and their content is lost upon a system-wide failure [3].

Summarizing, the main contributions of this paper are: i) we propose ISB-tracking, a new mechanical transformation for deriving detectably recoverable implementations of concurrent data structures, ii) in a system with volatile caches, we present how persistency instructions can be added in ISB-tracking in a manner that enhances efficiency and scalability, iii) we apply ISB-tracking to get new detectably recoverable implementations of a wide collection of data structures, and iv) we provide an experimental analysis to compare ISB tracking with *all* existing relevant transformations and detectably recoverable concurrent data structures we are aware of. They show the feasibility of ISB-tracking and the good scalability it exhibits in many cases.

*Full version available at: http://arxiv.org/abs/1905.13600.
†Supported in part by ISF grant 380/18.
‡This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme «Human Resources Development, Education and Lifelong Learning» in the context of the project "Reinforcement of Postdoctoral Researchers - 2nd Cycle" (MIS-5033021), implemented by the State Scholarships Foundation (IKY).

REFERENCES

[1] Hagit Attiya, Ohad Ben-Baruch, and Danny Hendler. 2018. Nesting-Safe Recoverable Linearizability: Modular Constructions for Non-Volatile Memory. In *ACM Symp on Principles of Distributed Computing (PODC)*. 7–16.
[2] Michal Friedman, Maurice Herlihy, Virendra J. Marathe, and Erez Petrank. 2018. A persistent lock-free queue for non-volatile memory. In *23rd ACM SIGPLAN Symp on Principles and Practice of Parallel Programming (PPoPP)*. 28–40.
[3] Joseph Izraelevitz, Hammurabi Mendes, and Michael L. Scott. 2016. Linearizability of Persistent Memory Objects Under a Full-System-Crash Failure Model. In *30th International Symp on Distributed Computing DISC*. 313–327.