Fast Concurrent Data-Structures Through Explicit Timestamping

Mike Dodds, Andreas Haas, Christoph M. Kirsch
mike.dodds@york.ac.uk, ahaas@cs.uni-salzburg.at, ck@cs.uni-salzburg.at

Abstract
Concurrent data-structures, such as stacks, queues and deques, often implicitly enforce a total order over elements with their underlying memory layout. However, linearizability only requires that elements are ordered if the inserting methods ran sequentially. We propose a new data-structure design which uses explicit timestamping to avoid unwanted ordering. Elements can be left unordered by associating them with unordered timestamps if their insert operations ran concurrently. In our approach, more concur-

Key Ideas
- Order elements in the data-structure only partially by using explicit timestamping.
- Store elements in thread-local buffers to avoid synchronization of insert operations.
- Use the RDTSCP CPU instruction for highly-scalable timestamp generation.
- Use intervals as timestamps to reduce the order on timestamps while still providing linearizability.

TS Deque Pseudo Code

```java
TS_Deque {  
    TS_Buffer buffer;
    void insertR(Element element) {  
        item = buffer.insertR(element);
        timestamp = buffer.newTimestamp();
        buffer.setTimestamp(item, timestamp);
    }
    Element removeR() {  
        do {  
            item = buffer.removeR();
        } while (!item.isValid());
        if (item.isEmpty())
            return EMPTY;
        else
            return item.element;
    }
}
```

Correctness
- The TS deque is linearizable with respect to the sequential specification of a deque.
- The remove operations of the TS deque are lock-free, the insert operations are wait-

Additional Information
http://scal.cs.uni-salzburg.at/tsstack

Acknowledgements
This work has been supported by the National Research Network RISE on Rigorous Systems Engineering (Austrian Science Fund (FWF): S11404-N23).