A concurrent implementation of a binary search tree dictionary

Diem Hoang Nguyen - hong@itu.dk
Mustapha Malik Bekkouche - mube@itu.dk
Supervisor: Peter Sestoft

January 2018

1 Abstract - please ignore

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam commodo neque a urna rutrum, quis viverra libero vehicula. Vivamus aliquam ligula nec magna venenatis dictum. Curabitur vel suscipit ligula, in lacinia ligula. Morbi sed sem sollicitudin, pulvinar erat non, vulputate mauris. Integer pharetra, ligula quis mattis aliquet, erat nibh cursus magna, maximus ullamcorper mauris odio et nunc. Morbi et risus interdum, volutpat enim posuere, pulvinar quam. Mauris vestibulum, lectus quis accumsan posuere, ligula urna tincidunt quam, at cursus lectus leo at dolor. Nulla sit amet nisi purus. Nunc non diam tristique, feugiat libero at, viverra tellus. Sed et lacus odio. Maecenas non tellus nec ante porta molestie in et ligula. Aliquam urna ante, sagittis vitae fermentum ac, lacinia in dolor. Ut id pretium massa, a pharetra lorem. Vestibulum sit amet blandit nisi. Sed volutpat in arcu et euismod. Suspendisse in sem at nulla blandit lobortis sed eu odio.

Sed porta et sapien vitae placerat. Suspendisse at cursus elit. Donec nec pellentesque sem. Donec vitae metus sit amet turpis scelerisque aliquet. Nunc viverra nec velit ac interdum. Nullam hendrerit sapien ligula, quis pulvinar lorem hendrerit quis. Curabitur et fringilla tortor. Quisque scelerisque interdum ligula, quis mattis sapien vehicula at. Sed fermentum erat ut tristique volutpat. Ut vitae interdum magna.

Contents

2 Introduction - please ignore

For years, processor manufacturers delivered increases in clock rates, so that single-threaded code executed faster on newer processors with no modification. As we are increasingly approaching the limits of Moore's law with transistors getting as small as 5 nm each, it has become harder to scale up clock rates without negative side effects. Constructors have thus turned to multi-core architectures for processors, this architecture allows parallelization of execution. With this parallel paradigm comes benefits and drawbacks, while it can allow a huge increase in performance and throughput it also increases the possibility for developers of making mistakes because of the unpredictable execution flow and the necessity to use specially designed classes for this paradigm called thread safe classes.

3 Background

3.1 Tree dictionaries

3.2 Concurrent tree dictionaries

3.3 Brown's work (working title)

4 Current concurrency tree dictionary implementations

5 Tests

5.1 Tests for correctness

5.2 Tests for scalability

6 Test results

7 Concurrent tree dictionary

7.1 Mark I

7.2 Mark II

7.3 Mark III

7.4 ... etc

7.5 further improvements

8 Discussion

9 Conclusion

10 References