# Exploring the Scalability Benefits of Relaxed Structural Invariance

*Joon Choi*
*Vincent Gramoli*
School of Information Technologies

## INTRODUCTION

### Concurrent Data Structures

- Trends and advancements in hardware has multicore processors increasing their number of cores every year.
- To maximise performance gain from these processors, research and development of concurrent data structures which handle contention better has been a hot topic in the parallel computing scene.
- Many of these data structures rely on their Structural Invariance to keep good access times.
- Under high contention, upholding the Structural Invariance is typically costly. This research attempts to find benefits of relaxing it to a point which yields better scalability and performance.

### Structural Invariance

- Structural Invariance (SI) of data structures are important and necessary to data structures in order to promise time complexities.
- For example, Trees have logarithmic access times given that their SI is kept through restructuring and therefore staying ‘balanced’.
- Keeping a data structure’s SI usually involves locking or attaining large portions of the data structures. In highly concurrent settings where multiple threads are already contending for data, operations enforcing SI typically leads to more contention resulting in poorer scalability [1]. See Figure 1.
- Recent research has focused on providing concurrent data structures, which minimise contention caused by methods that enforce their SI.

### Speculation Friendly BST [2]

- Speculation Friendly BST (SFTree) is a state-of-art binary search tree which minimises contention from keeping its SI.
- It decouples insertions and deletions of nodes with restructuring of the tree to keep its SI which results in state-of-art scalability and performance.
- This reduces contention by finishing the execution of insert and remove operations faster (as they don’t trigger restructuring) and use a dedicated maintenance thread to do the restructuring.
- This effectively lowers the chance of conflicts leading to a decrease in contention and a better performance by achieving speeds 3.5 faster than other available data structures when used on transaction based travel reservation application.
- However, the dedicated thread creates interface issues and usage of SFTree involves spawning and management of an extra thread, taking much away from its usability and applicability.

## THE CONTRIBUTION

- To explore and confirm that relaxed SI increases scalability of concurrent data structures, using SFTree as a representative of a state-of-art concurrent data structure.
- Changes to SFTree were necessary to benchmark and research benefits of relaxed SI:
  - Recouple restructuring with insertions and deletions (similar to the avl trees which SFTree was succeeding) to abolish the need to spawn and manage dedicated maintenance thread.
  - Control how often the restructuring takes place through probabilistically running the restructuring hence controlling how tight/relaxed the enforcement of SI is (lower the probability, the more relaxed).
  - Benchmark and find a probability (degree of relaxedness) which yields optimum scalability and performance especially in high contention settings.

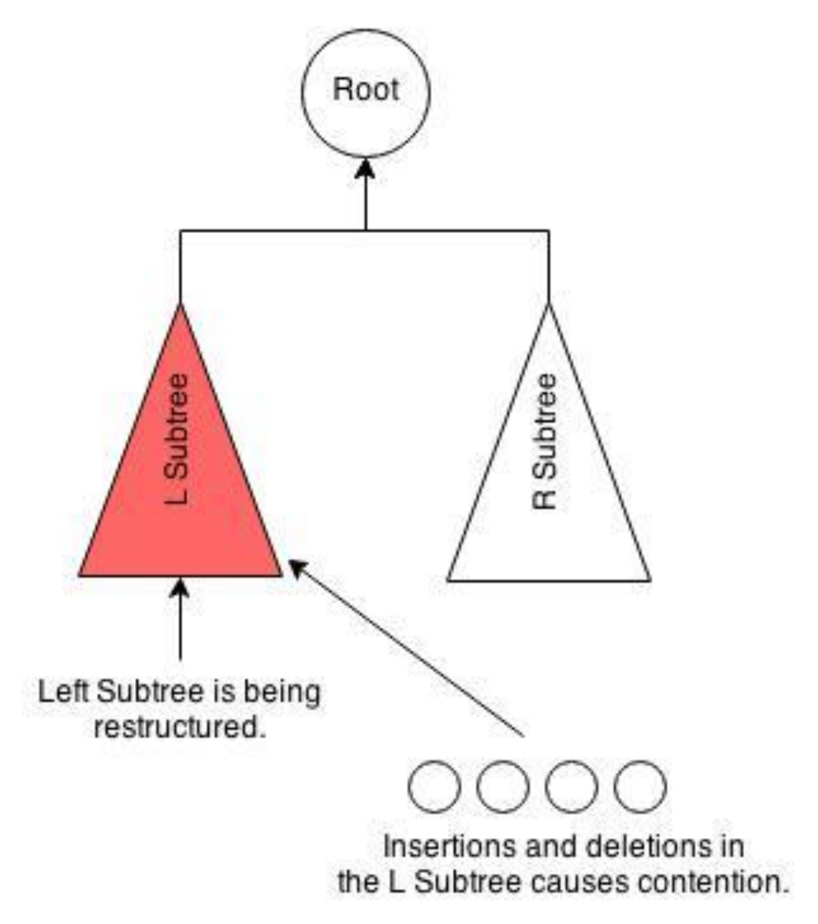
Root
L Subtree
R Subtree
Left Subtree is being restructured.
Insertions and deletions in the L Subtree causes contention.

***Figure 1: How enforcing SI can cause contention.***

## THE EXPERIMENT

- The changes to SFTree were implemented on C programming language.
- Transaction throughput was documented using Syncrobench benchmarking tool.
- Different parameters were manipulated in the benchmarks to see which probabilities gained or lost performance and scalability under different scenarios.

## THE RESULTS (see Figure 2)

- Tests which had probability of running restructuring after insertions and deletions which were close to 0 performed considerably worse as the number of threads increased, showing that the SI is necessary.
- However, as the probability of restructuring increased, scalability is greatly reduced.
- The experiment showed that relaxing the SI on the SFTree to a certain degree benefits performance and scalability (middle graph of Figure 2) and that there are ‘sweet spot’ probabilities which yields optimum scalability and performance.

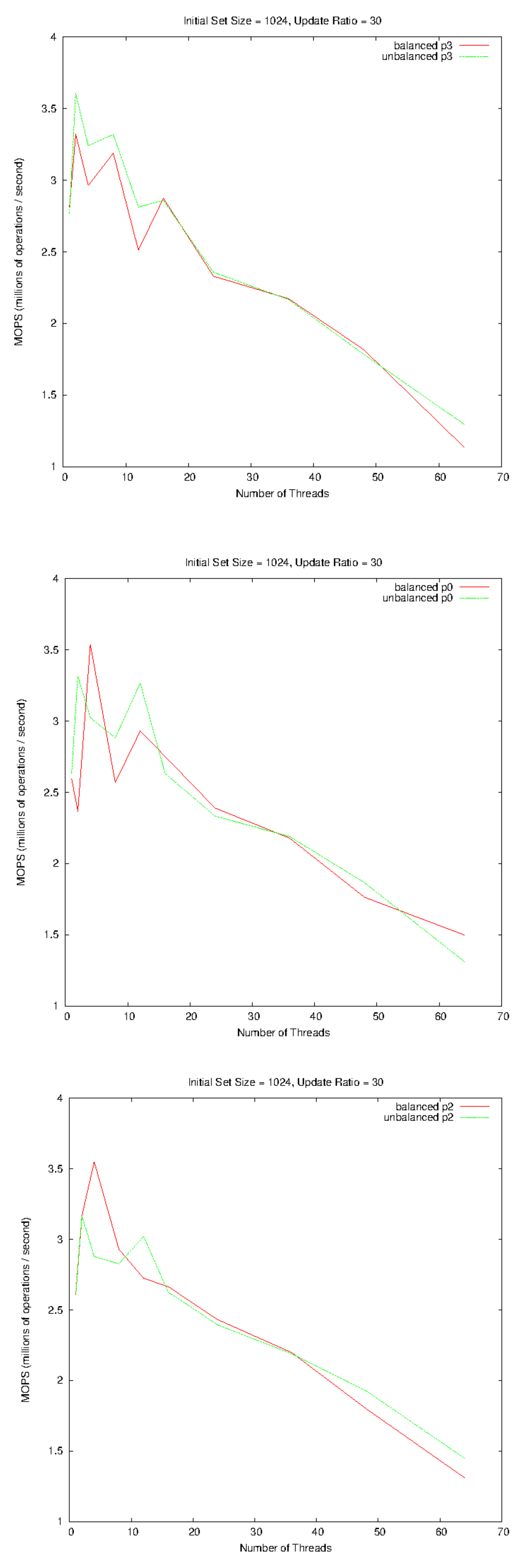

***Figure 2: Performance experiments showing top (too low probability), middle (good probability), bottom (too high).***

## FUTURE WORK

- Apply the relaxed SI Concept to other contemporary concurrent data structures which rely on restructuring or maintenance to keep its SI such as the No Hot Spot Rotating Skip List.
- Examine the optimum probabilities across data structures and look for patterns.

## REFERENCES

1. K. Larsen. " AVL Trees with relaxed balance", in proc. of the 8th International symp. on Parallel Processing, 1994
2. V. Gramoli .et al. "A Speculation-Friendly Binary Search Tree", in proc. of the 17th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming, 2012.

THIS RESEARCH IS SPONSORED BY

Insert SPONSOR LOGO