FINE TUNING THE ROTATING SKIP LIST

Itamar Talmon, Tal Shoef

Tel Aviv University

ABSTRACT

Skip list [1] is a linked-list-based structure that became an increasingly popular concurrent alternative to search trees due to its logerithmic complexity and local balancing operation. The Rotating skip list [2] is the fastest concurrent skip list to date. In this paper, we investigate, and try to improve upon, the different heuristics used in the Rotating skip list data structure.

1. INTRODUCTION

1.1. Concurrent Skip Lists

Skip list is a mashu mashu

1.2. Our Contribution

We examined and extended the following heuristics:

- Level Hight Heuristics
- Level Delete Heuristics
- Deletion Help Heuristics
- Background Thread Sleeping Heuristics

Detailed descriptions of the changes made, in addition to the evaluations, are included in the prociding sections of the paper.

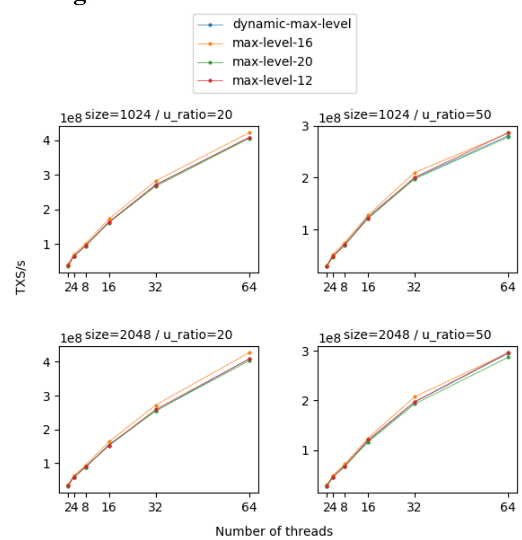
1.3. Code Location

The complete code, including the compiled data strucures and the evaluation script, could be find at `github.com/itamartalmon/synchrobench`.

2. LEVEL HIGHT HEURISTICS

The Maximum Level of the data structure , BLABLABLA

Fig. 1. Max Level Heuristics Performence



2.1. Fixed Max Level

Fixing the Max level is commonly use, blablblal

As can be seen on figure BLABLALB, ads bla

2.2. Dynamic Max Level

Dynamic Max level is more blablblal

2.2.1. Evaluation

As can be seen on figure bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

bli bla blu

Fig. 2. Tall Delelted By Size Deletion Performence

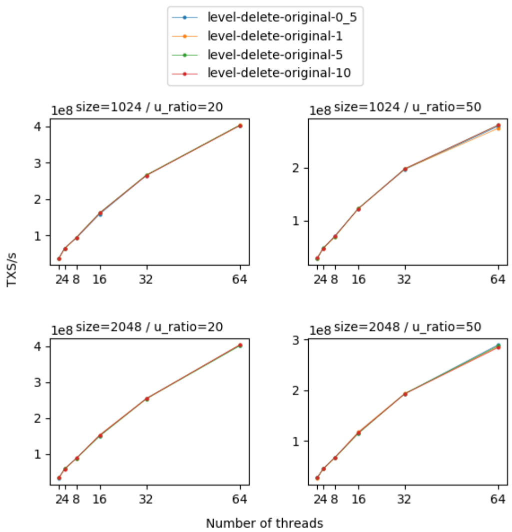
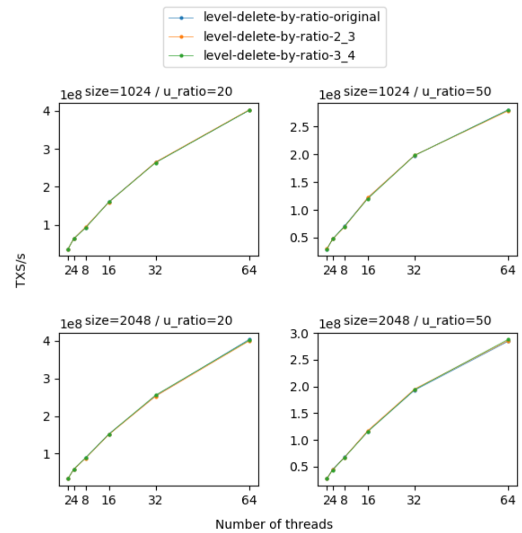


Fig. 3. Levels Ratio Deletion Performence



3. ROTAION DELETION HEURISTICS

When deleting blablabae , BLABLABLA

3.1. Tall-Deletions-Size Rate

In the original paper use, blablblal

3.1.1. Evaluation

As can be seen on figure BLABLALB, ads bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

3.2. Relative Level Size

Dynamic Max level is more blablblal

3.2.1. Evaluation

As can be seen on figure BLABLALB, ads bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

4. DELETION HELP HEURISTICS

4.1. Thread-Num-Size Rate

Dynamic Max level is more blablblal

4.1.1. Evaluation

As can be seen on figure BLABLALB, ads bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla

5. BACKGROUND THREAD SLEEP HEURISTICS

The background blablabae , BLABLABLA

5.1. Thread-Num Rate

In the original paper use, blablblal

5.1.1. Evaluation

As can be seen on figure BLABLALB, ads

6. EXPERIMENTAL SETTINGS

In this section we describe the multicore machines and the benchmarks used in our experiments as well as the 7 data structure algorithms we compare our rotating skip list against. Multicore machines. We used two different multicore machines to validate our results, one with 2 8-way Intel Xeon E5-2450 processors with hyperthreading... BLABLA

7. FUTURE RESEARCH

BLIA BALIDSA ADBLADS

Fig. 4. Help-Remove Performence

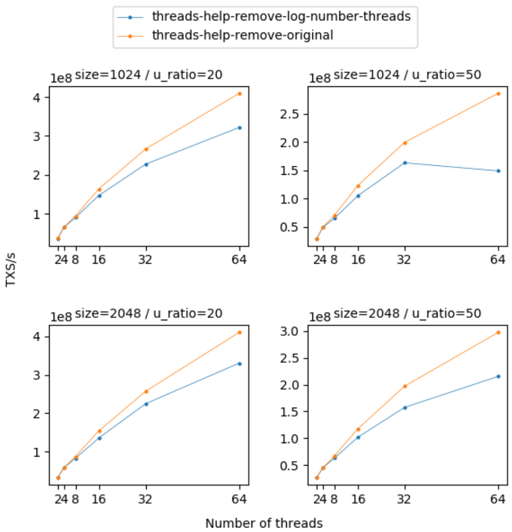
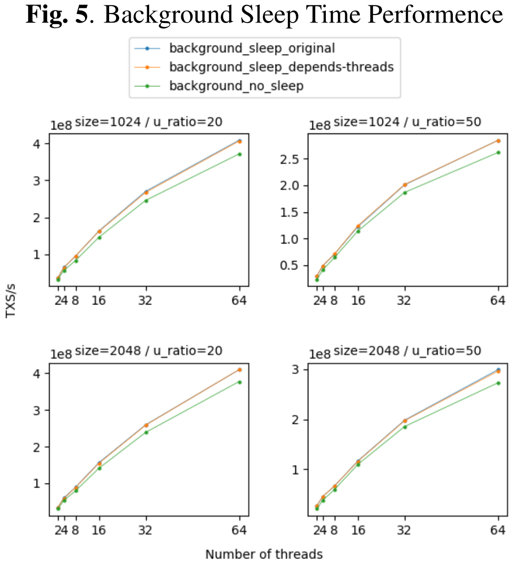


Fig. 5. Background Sleep Time Performence



8. REFERENCES

[1] William Pugh, “Skip lists: A probabilistic alternative to balanced trees,” in *Workshop on Algorithms and Data Structures*, 1989, vol. August, pp. 437–449.

[2] Alan Fekete Dick, Ian and Vincent Gramoli, “A skip list for multicore,” in *Concurrency and Computation: Practice and Experience*, 2017, vol. 29, p. 4.

[3] Gramoli Vincent Crain, Tyler and Michel Raynal, “No hot spot non-blocking skip list,” in *Distributed Computing Systems (ICDCS)*, 2013, vol. 33, pp. 196–205.