Skip List Optimisation: Finding the Best Height

Introduction

Skip lists are a probabilistic data structure that allows efficient search, insertion, and deletion operations. They are particularly useful for applications that require fast search operations, such as spell-checkers, database indexing, and routing tables in computer networks. This report presents an analysis of the performance characteristics of skip lists, focusing on the optimal height that provides the best trade-off between speed and memory usage.

> ⚠️ **Note on Data Collection**
>
> Please be advised that the data presented in this report was collected on a 16-inch, 2021 MacBook Pro with an Apple M1 Pro processor and 16 GB of Memory, running macOS Ventura 13.2.1 and virtualising Ubuntu 22.10. It is important to note that the performance results may vary depending on the system and configurations. Additionally, the algorithm runs were performed single-threaded.

Methodology

To find the best height, the following, timed approach, is employed:

1. Iterate over a range of possible heights (this report accounts for an iteration from 5 to 100).
2. For each height, create a skip list with that maximum height.
3. Load a dictionary file into the skip list and then search for errors in a text file using that skip list.
4. Measure the execution time for each height.
5. Compare the execution times for all the heights and select the one with the lowest time.

Performance Analysis

When starting with a low height, such as 4, it is observed that the skip list gets exponentially faster as the height is increased. This can be attributed to the fact that with a higher height, the skip list can skip more elements in a single step, leading to faster search operations.

As the height continues to increase, the improvements in execution time become less significant. When a height of 10 is reached, the performance gains start to diminish. From height 10 to 15, there is still a noticeable improvement in speed, but the increments are smaller compared to the previous heights.

It is thus found that the optimal height lies around 18, considering a margin of 2 values. This means that the best performance is achieved when the maximum height is set between 16 and 20.

Upon surpassing the optimal height value, the benefits of increased height begin to diminish, or even disappear completely. This is primarily because the memory overhead associated with the additional pointers required for each level increases, negatively impacting the overall performance. Additionally, the advantage of skipping nodes at higher levels becomes less significant due to the reduced probability of nodes having pointers at these levels. Consequently, the performance gains are offset by the increased memory usage and pointer traversal overhead, leading to diminishing returns.

Conclusion

Through this analysis, the key factors that contribute to the performance of a skip list have been identified. By carefully selecting the appropriate height, the search operations can be optimised, and the benefits of the skip list data structure can be fully leveraged. The findings demonstrate that the optimal height lies in the range of 16 to 20, with the most significant performance gains occurring when increasing the height up to 10. Beyond this point, the improvements become less pronounced but still contribute to the overall efficiency. Understanding these nuances enables the fine-tuning of the skip list for optimal performance in various applications. However, it is crucial to be aware that exceeding the optimal height may lead to diminishing returns and increased memory usage.