CS 5510 Project: Concurrent Network Cache

Abhishek Chauhan
zxcve@vt.edu

VirginiaTech
Invent the Future

William Naciri
applewil@vt.edu

Introduction

- A cache is a hardware or software component that stores data so future requests can be served faster.

Motivation

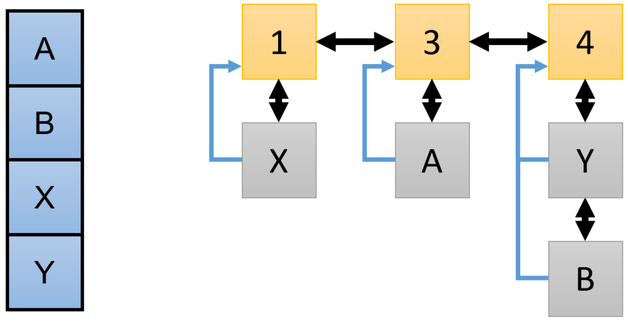
- The benefits of a network cache are reduced bandwidth, server load and perceived lag.

Implementation

- Global cache with least frequently used eviction policy.
- Advantageous for web browsing and provides O(1) run time complexity.

Data Structures	
Hashtable	• Pointers for accessing data nodes
Parent Node	• Doubly linked list of different frequencies
Data Node	• Doubly linked list of cached elements • Tracks parent pointers

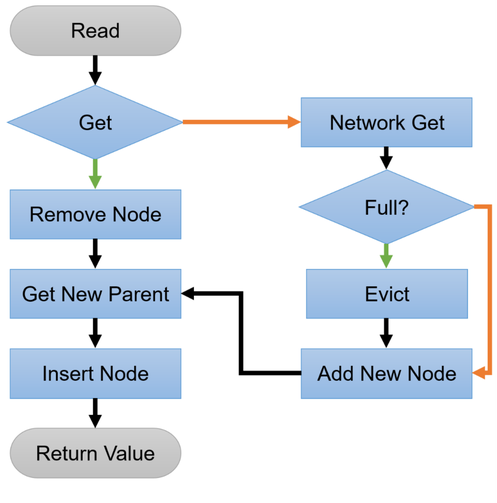
Figure 1: LFU cache



Approach

- We based our project on a sequential least frequently used cache.
- To extend this design to support concurrent access we converted all data structures and synchronized their operations.
- The upgraded cache uses our ConcurrentDoublyLinkedList and Java's ConcurrentHashMap.
- All functionality is encapsulated inside one method, read.

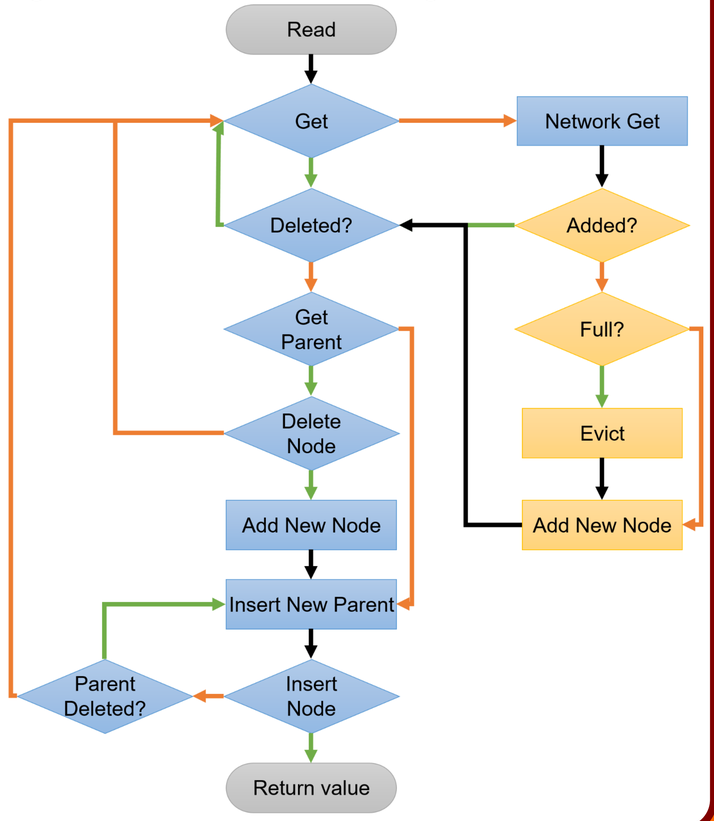
Figure 2: Sequential cache algorithm



Concurrent Cache

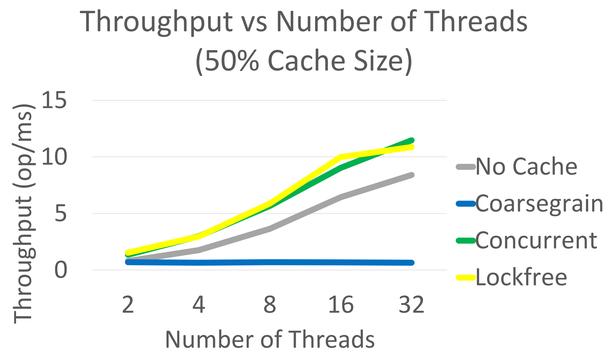
- The solution is nearly lockfree. Each of the blue procedures are independently lockfree.
- The linearization point of read is when *Insert Node* returns true.
- The algorithm is not linearizable for the case when multiple threads insert into a full cache. For this we have added a synchronized insert & evict in yellow.
- There are two cycles that can incur starvation. Nonetheless, the algorithm guarantees at least one thread makes progress.

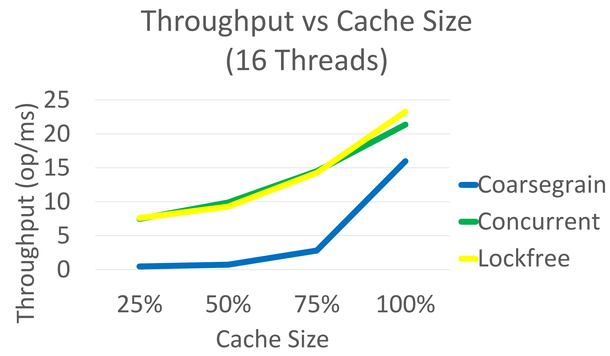
Figure 3: Concurrent cache algorithm



Results

Figure 4: Charts





Discussion

- Considering we achieved a near lockfree implementation we decided to remove the synchronize block to evaluate a non-linearizable lockfree solution. This implementation observed slightly better performance.
- Not only does our concurrent solution provide correct parallel execution but it observed comparable performance to the lockfree implementation.
- The tradeoff for correctness over performance is desirable.