Software Engineering Intern – Web APIs

Amir Assadollahzadeh
July 26,17 – Sep 20, 17
San Diego, CA

INTRO

This summer I learned how a large scale website works and the amount of work that makes it run. I worked on reducing GraphQL Schema footprint in memory.

Key Achievement

During my Internship at ServiceNow we needed to make changes to the GraphQL Java API to reduce our memory footprint since we only have 2GB of RAM. I was tasked with Implementing a Hash Map that would be space and time efficient. I implemented two different Hash Maps (**Open Addressing & Cuckoo map**). I developed these two algorithms using TDD, using this method to implement both algorithms really helped me find bugs that I would introduce to my code as I kept adding more functionality. The Open Addressing map had an average of look up of ~2 for 80% load factor. But we needed to save more memory! Doing some research let me to implement a Cuckoo map and I learned that a 5-array Cuckoo map will result in 98% fill factor with at most 5 lookups! We ended up using a hybrid approach with a different Hash Map (ArrayMap) that ran a binary search on insert and lookup. In this approach approach smaller and less popular data used the ArrayMap and the larger and popular data used the Cuckoo map. We tested the algorithm on a real customer instance and saw a ~67% reduction in our memory footprint using a Java Profiler.

Connecting to UCSD

My experience at UCSD really helped me excel in my internship and be able to contribute my skills to the team. I feel that UCSD gave me a very good foundation. I was part of the Autograder (autograder.ucsd.edu) team at UCSD for 6 quarters and I have learned extremely valuable knowledge that I was able to apply to my internship. Some things I learned in Autograder and applied to my internship:

- Technologies, Frameworks, Design
    - Intelij
    - SQL
    - Java
    - Designing features so that they are maintainable and extendible
- Working in a team environment
    - Being able to communicate effectively so the team is functional
    - Being able to work in a pair and be productive
    - Explain the reason for writing a piece of code
    - Creating a connection with the team

I also tutored for CSE 12 & 15L for 6 quarters tutoring really helped me in

- Being able to solve solutions that I immediately don't have answers to
- Being able to explain something effectively so that the other person understand
- Reading through different peoples code and figuring out what is going wrong
- Strengthen my understanding in basic data structures

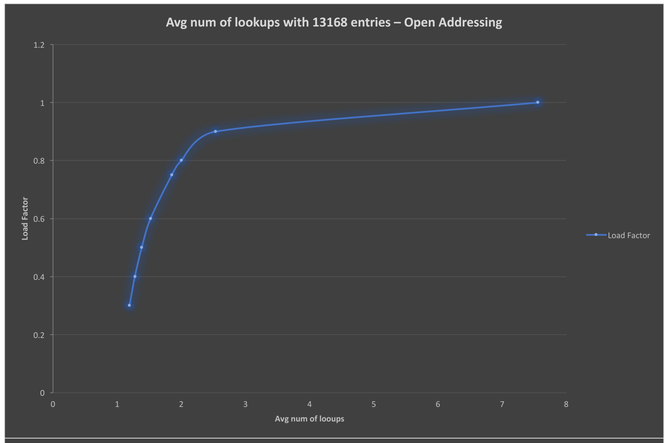
Take Away

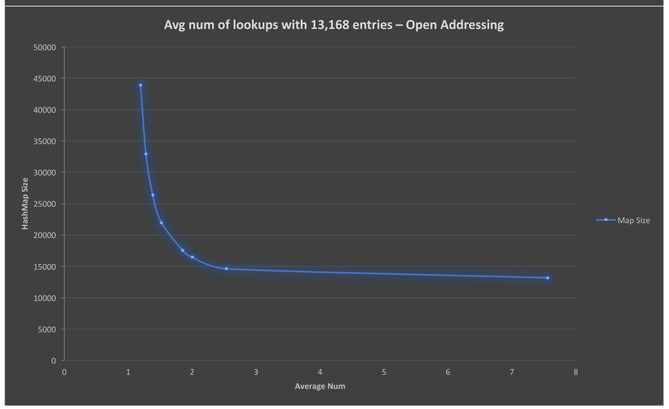
This internship really showed me how much I enjoy studying Computer Science. I had always dreamed about having a job where I am excited to wake up in the morning and go to work. I had to stop taking my computer home so I don't keep working! My team was very welcoming and helpful, at times I did not think I was an intern. it always felt like I was part of the team, working on features that will effect many people using the product. I was able to make relationships that I think will last for a very time time.

What I learned

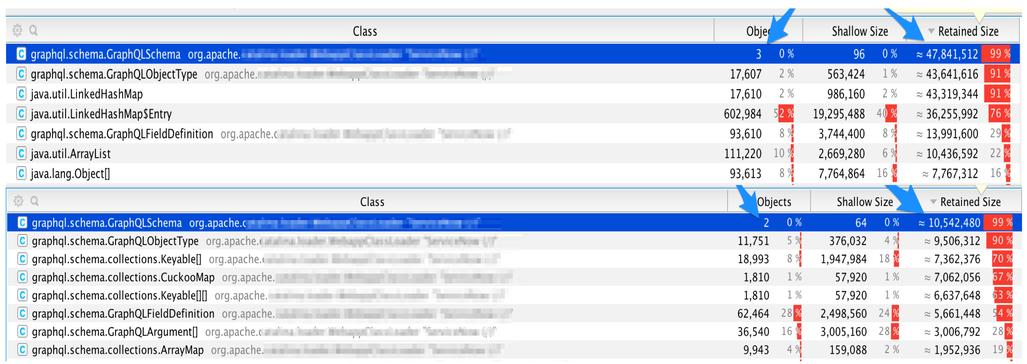
- Java 8
- Unit Testing, It Testing
    - Power Mock
    - Mockito
- Java Profiling
- TDD
- Extreme Programming
- Design Patterns for more readable code
- InteliJ Features
- How to go about designing code
    - If unsure about which way to implement something then just do the simplest case and make it work. Once it is doing what you expect then you can begin to make bigger design decisions. Less time is wasted on making a decision

Performance of Open Addressing





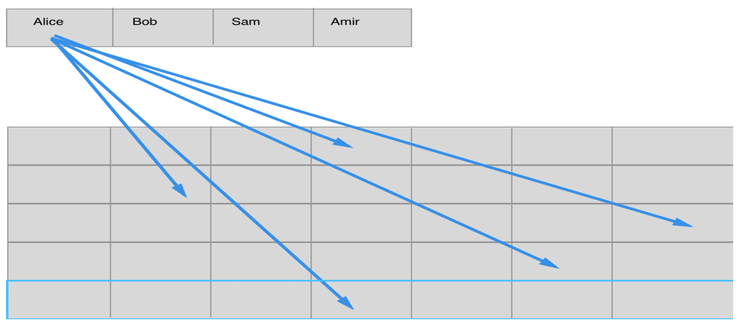
Before & After results of Cuckoo map memory reduction



Class	Objects		Shallow Size		Retained Size	
graphql.schema.GraphQLSchema org.apache.	3	0 %	96	0 %	≈ 47,841,512	99 %
graphql.schema.GraphQLObjectType org.apache.	17,607	2 %	563,424	1 %	≈ 43,641,616	91 %
java.util.LinkedHashMap	17,610	2 %	986,160	2 %	≈ 43,319,344	91 %
java.util.LinkedHashMap$Entry	602,984	52 %	19,295,488	40 %	≈ 36,255,992	76 %
graphql.schema.GraphQLFieldDefinition org.apache.c	93,610	8 %	3,744,400	8 %	≈ 13,991,600	29 %
java.util.ArrayList	111,220	10 %	2,669,280	6 %	≈ 10,436,592	22 %
java.lang.Object[]	93,613	8 %	7,764,864	16 %	≈ 7,767,312	16 %

Class	Objects		Shallow Size		Retained Size	
graphql.schema.GraphQLSchema org.apache.c	2	0 %	64	0 %	≈ 10,542,480	99 %
graphql.schema.GraphQLObjectType org.apache.c	11,751	5 %	376,032	4 %	≈ 9,506,312	90 %
graphql.schema.collections.Keyable[] org.apache.c	18,993	8 %	1,947,984	18 %	≈ 7,362,376	70 %
graphql.schema.collections.CuckooMap org.apache.	1,810	1 %	57,920	1 %	≈ 7,062,056	67 %
graphql.schema.collections.Keyable[][] org.apache.	1,810	1 %	57,920	1 %	≈ 6,637,648	63 %
graphql.schema.GraphQLFieldDefinition org.apache.	62,464	28 %	2,498,560	24 %	≈ 5,661,448	54 %
graphql.schema.GraphQLArgument[] org.apache.c	36,540	16 %	3,005,160	28 %	≈ 3,006,792	28 %
graphql.schema.collections.ArrayMap org.apache.	9,943	4 %	159,088	2 %	≈ 1,952,936	19 %

5-array Cuckoo Map



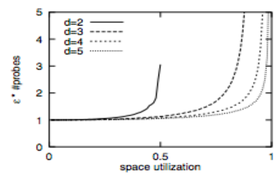


Fig. 2. Scaled average number of memory probes for insertion into a d-ary Cuckoo Hash table with 100 000 entries as a function of the memory utilization $n/10^5$ ($\epsilon = 1 - n/10^5$). Starting from $n = 1000 \cdot k$ ($k \in \{1, \ldots, 100\}$), a random element is removed and a new random element is inserted. This is repeated 1000 times for each of 100 independent runs. Hash functions are full lookup tables filled with random elements generated using [18]. The curves stop when any insertion fails after 1000 probes.

and grow quickly as they approach a capacity threshold that depends on d. Increasing d strictly decreases average insertion time so that we get clear trade-off between worst case number of probes for accesses and average insertion time.

The maximum space utilization approaches one quickly as d is incremented. The observed thresholds were at 49 % for $d = 2$, 91 % at $d = 3$, 97 % at $d = 4$, and 99 % at $d = 5$.