System Design and Innovation: A Garbage Collection Case Study — Kathryn S McKinley

Lecture 2 - June 27, 2025

1 A Brief Introduction to Concurrency

We will continue to build upon our previous lecture by discussing concurrent and touching upon parallel garbage collectors. The first garbage collector to implement concurrency is the c4 garbage collector. Concurrent garbage collection minimizes pauses by taking a snapshot of the world and running while the program is running. Having a snapshot at the beginning is critical to all concurrent algorithms.

Concurrent garbage collection isn't perfect. Take Shenandoah and G1 for example, these are 2 concurrent garbage collectors that are implemented differently. Shenandoah has pauses that are much shorter than those of G1, but with the drawback of having much greater latency.

2 A Brief Insight into LXR Design

LXR is a high throughput garbage collector that is based on reference counting with no concurrent copying. It has low latency due to taking short but frequent pauses and taking snapshots at the beginning. LXR took about 5 years of work, but the end result is a garbage collector with similar latency as the G1 garbage collector but with much shorter pauses. While this does make LXR a strong alternative to G1, it has the downside of being quite complicated.

3 A Brief Summary of Industry JVM Collectors

- G1 (53k lines of code)
- C4 (40k lines of code)
- Shenandoah (33k lines of code)
- ZGC (66k lines of code)

All of the above garbage collectors have a number of caveats, namely they have low code reuse (take the g1 garbage collector for example, while it does have 38k lines' worth of shared code, these 38k lines are on top of the 53k lines of the G1 garbage collector), low maintability (most of these garbage collectors have not been updated in 2 years), and low performance.

4 A Brief Introduction to Work Packets

Work packets are a new form of abstraction at the lowest level of the garbage collector that identify and mark objects that are still in use by the program. They do this by tracing all object reference chains from the roots. One of the main advantages of using work packets is that they introduced a layer of abstraction which led to improvements in parallelism and software engineering. To this extent, we will be comparing garbage collection with and without work packets.

4.1 Basic GC phases

The basic garbage collector phases consist of

1. stop mutators (programs)
2. reset mark table (determining what is dead and what is alive)
3. scan roots (pointing to the heap)
4. transitive closure
5. sweep blocks
6. resume mutators

It should be noted that steps 2 to 4 are all performed sequentially, with the main drawback being that this implementation has low parallelism and poor reusability. We will be treating immix with phases as if it had the same phases as a basic garbage collector in our high-level overview.

4.2 Immix with work packets and dependencies

The basic phases of immix with work packets and dependencies are mostly the same as immix with phases, but with the caveat of adding parallelism and dependent based scheduling, So for example, the phases might look something like:

1. stop mutators
2. reset mark table & (scan roots + transitive closure) in parallel
3. sweep blocks
4. resume mutators

By step 2, we mean here that resetting mark table can be done in parallel to scanning roots and finding the transitive closure of references.

4.3 Evaluation

Most components tend to be reusable. This is evident when observing that 50 % of the workspaces used are common work packets. In addition, garbage collectors need only declare their packets and dependencies. Some of the common work packets include

- root scanning
- marking and evacuation
- mark table zeroing
- sweeping
- stop/resume mutators

4.4 Lessons for Parallel Programming

After summarizing the last 25 years of innovation and design in the field of garbage collection, we should note that there are still new abstractions to implement, and that their benefits include efficiency, faster innovation, handling different types of work with dependencies, and reusability.

5 Better Computing Culture with a Research Mindset = More Innovation

Computer Science, like any field, is filled with a plethora of myths and misconceptions. We believe it is essential to know the realities of the field so that we can distance ourselves from these misconceptions and so that we can have a better research culture and mindset.

It is commonly perceived that everything emanated from the isolated founding fathers of Silicon Valley. This is not the case, as each of the founding fathers had a partner throughout their journey. For leadership and results aren't simply concentrated in a single person, rather they are built by a group of people collaborating and building on top of one another's ideas. For example, Steve Jobs had Steve Wozniak, Bill Gates had Paul Allen, Larry Page had Sergey Brin, and the list just keeps going.

The fact of the matter is, collaboration is necessary to achieve any real results. For we are standing on the shoulders of giants, and collaborating with others will lead us to new advancements and discoveries.

As noted by Blackburn, McKinley, and Xie in 2019, The number of ACM authors per paper at some of the top labs in the country has been rising steadily over the years. This was also the case in Woolley and Malone's 2010 paper which has revealed that the average social sensitivity of the group, the equality of turns in the conversation, and the number of women in the group were the main predictors of team performance as opposed to the average or the top IQ.

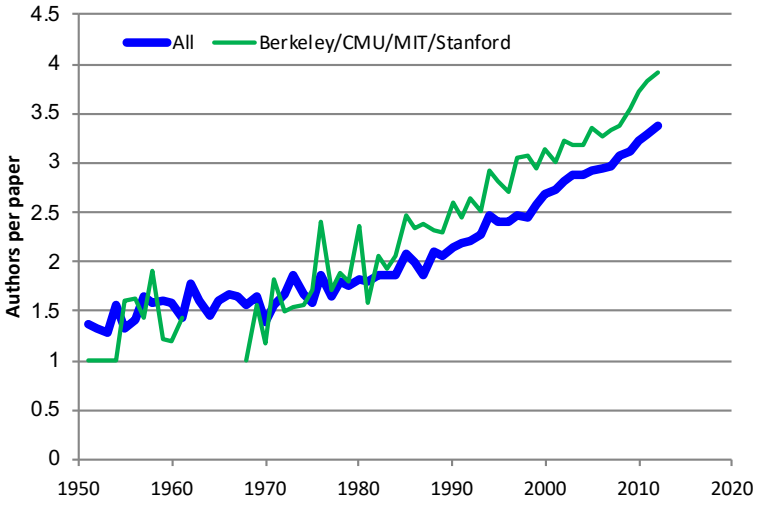

Figure 1: Average Number of ACM Authors from [BMX19]

One other thing to note is that one should be resilient to failure during their research and PhD journey, for failure is bound to happen. Whether it is in the form of a paper getting rejected for the first time, or a paper never getting accepted, failure will happen, and it is then that one's interpersonal relationships will help them with the stresses and failures experienced throughout grad school, research, and life in general.

References

[BMX19] Stephen M. Blackburn, Kathryn S. McKinley, and Lexing Xie. *Author Growth Outstrips Publication Growth in Computer Science and Publication Quality Correlates with Collaboration*. 2019. arXiv: 1909.02212 [cs.DL]. URL: https://arxiv.org/abs/1909.02212.