UC SANTA BARBARA

Project Vision and Plan

Vision

The Go garbage collector implements a concurrent, tri-color, mark and sweep algorithm that is triggered periodically or based on memory pressure (heap target has been exceeded). Although the Go garbage collector provides a knob GOGC to adjust the size of the total heap relative to the size of live objects, applications that see frequent and/or high variations in load can still struggle with high CPU usage from GC cycles and possibly even OOMs due to the GC being too aggressive or too passive. We would like to investigate the intricacies of Go's garbage collector, hoping to improve on its runtime pacer by leveraging the GOGC knob dynamically along with other GC knobs such as GOMEMLIMIT.

Plan

- Learn more about the Golang Garbage Collector
    - Memory Allocation Patterns, Escape Analysis, Concurrent Tri-Color Mark and Sweep
- Benchmark 3 different programs in Go with the default settings of GOGC
    - Use pprof
- Play with GC knobs such as GOGC and GOMEMLIMIT
    - Read papers from Uber, Weaviate, Twitch on ways to mitigate high CPU usage and OOMs caused by Go's garbage collection
- Implement a variation of these solutions and compare with our previous benchmark results on the same programs

Resources

- https://www.uber.com/blog/how-we-saved-70k-cores-across-30-mission-critical-services/
- https://weaviate.io/blog/gomemlimit-a-game-changer-for-high-memory-applications
- https://blog.twitch.tv/en/2019/04/10/go-memory-ballast-how-i-learnt-to-stop-worrying-and-love-the-heap/