CSA Parallel Report - James Forsters

Philosophy

The notable deviation I made from the given skeleton code is that I deleted it all and wrote everything from scratch. I chose only have one go routine/main thread that does both IO operations as well as GOL calculations, this simplifies the code as I didn't need any extra explicit channels or mutexes other than those handed to Run() (events, and keypresses)

To handle multiple threads, when you call **processOneTurnWithThreads()** (and some other functions) you give it the number of threads you want it to use, then within the function it splits the work over the given number of threads, waits till they all finish then returns the result.

This scope limited version of parallel processing keeps things that are meant to be parallel from those which don't need to be which means less overall thread safety concerns as there is less synchronization code overall (did I mention I didn't use any extra explicit channels 😀). And less synchronization code means less surface area for race conditions and deadlocks to grow.

Benchmark methodology

I used a barebones version of my code to run benchmarks on, called **bench.go**, which only worries itself with calculating the final turn and returns once it knows what it would be.
This is because of the extra features such as IO operations, events, keypresses, etc… muddy the water with things that aren't to do with the GOL implementation.

Independent Testing Variables

- **Machine**: Which computer you run the code on has an effect on the performance, not only the number or cores/threads the computer has but also the clock speeds, amount of cache, etc…
- **Turns**: How many turns do you simulate in the benchmark?
- **Size**: How large is the image used in the benchmark?
- **Image**: Which cells are on/off initially?
- **Threads**: How many threads do you tell the program it can use?

Testing Data

The next page contains the testing data for the **Cartesian Product** of the following sets of Independent Testing Variables:

- **Machine**: { "My Laptop (4-core)", "Random Lab Machine in 2.11 (6-core)" }
- **Turns**: { 0, 10, 100, 1000, 10000 }
- **Size**: { 256, 512 }
- **Image**: { "The image you gave us" }
- **Threads**: { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 }

“My Laptop (4-core)” Benchmarks

https://ibb.co/qDSkWFc (does this count as cheating the strict 6 page rule? Ah well 🙂)

“Random Lab Machine in 2.11 (6-core)” Benchmarks

https://ibb.co/CvjBgWt More graphs = Higher marks right? 😀

Benchmarks Analysis

Overvations:

- Generally more threads = less time but with diminishing returns
- My laptop performs best around multiples of 4 threads (4-cores)
- The lab machine performs best around 12 threads (6-core 12-threaded cpu)
- The lab machine is much stronger at multi-threaded processing compared to my laptop 😣