Concurrency: Maximizing minimal resources with architecture

Alex Hernandez

About me

- Sr Go Engineer @ Bluehalo
- Go fanboy
- Cars are cool
- I buy meme coins

Agenda

- Concurrency 101
- Go concurrency crash course
- Examples
- When it works/When it doesn’t
- Common Patterns
- Questions

Concurrency 101

Concurrency 101

- Go is a concurrent language.
  - Go routines
  - Channels
  - Select Statements
  - Synchronization

Concurrency 101

- Go is a concurrent language.
  - Go routines
  - Channels
  - Select Statements
  - Synchronization
- Concurrency is not parallelism.
  - Concurrency -
  - Parallelism -

Concurrency 101

- Go is a concurrent language.
  - Go routines
  - Channels
  - Select Statements
  - Synchronization support
- Concurrency is not parallelism.
  - Concurrency - dealing with multiple things at once (architecture)
  - Parallelism - doing multiple things at once (execution)

Concurrency 101

- Go is a concurrent language.
  - Go routines
  - Channels
  - Select Statements
  - Synchronization support
- Concurrency is not parallelism.
  - Concurrency - dealing with multiple things at once (architecture)
  - Parallelism - doing multiple things at once (execution)

Concurrency is a way to build our applications that **may** enable parallelism, but parallelism is not the goal of concurrency, the goal is good structure.

Concurrency 101

An analogy (borrowed from Rob Pike)

Concurrent: mouse, keyboard, display, disk driver, etc

Parallel: Sum of squares

Concurrency 101

Concurrency needs communication

Concurrency **is** architecture.

Concurrency is a way to structure a program by breaking it down into independently executing pieces.

Communication is the way we coordinate these independent pieces of code. (channels)

1978 - C.A.R Hoare: Communicating Sequential Processes

Go Concurrency Crash Course

Go Routines

- Heart of Go concurrency, everything revolves around the go routine.
- Lightweight - 2KB initial stack size, dynamic, can have thousands of go routines running at a time.
- Non-blocking - runs independently of main
- Managed by the Go scheduler.
- Easy - simply add the go keyword to run something as a go routine:
    - go myFunction()

Go Concurrency Crash Course

Channels

- Allows for communication between go routines
- Can be blocking or non-blocking (unbuffered/buffered)
- Can be direction specific (send-only/receive-only

Go Concurrency Crash Course

Select

- Mechanism to control program flow based on channels
- Can be blocking or non-blocking
- Can “listen” to n number of channels

Go Concurrency Crash Course

Examples

When it works/When it doesn’t

Works with I/O bound tasks, API calls, database, etc

Doesn’t work with algorithms/processing cpu bound tasks

Concurrency Patterns

Concurrency patterns are tried-and-true solutions for addressing common challenges in concurrent programming.

Concurrent Patterns offer several advantages:

- Base for more advanced patterns
- Reduce bugs (deadlocks, race conditions, etc)

Concurrency Patterns
Mutex Pattern

The Mutex Pattern protects shared resources with mutual exclusion locks, while also relying on

Select statements to allow access from various other patterns of the program.

Concurrency Patterns
Workerpool Pattern

- Used to limit the number of go routines spawned when processing large data sets.
- Prevents maxing out the CPU, while still maintaining benefits of concurrent and parallel processing.

Concurrency Patterns
Pipeline Pattern

- Used to connect a series of stages, where each stage consists of n concurrent functions, that process data and pass the results to the next stage in the pipeline.
- 3 stage pipeline:
  - Stage A: concurrent functions in stage A have no dependencies, do not rely on each other
  - Stage B: concurrent functions in stage B rely on information from stage A
  - Stage C: concurrent functions in stage C, rely on information from both stage A and stage B

Recap

- Concurrency is architecture! Build apps with concurrency in mind and reap the benefits of speed **without** having to pay for more hardware.
- Concurrency is **<u>not</u>** parallelism. Related, but distinct ideas.
- Go is a concurrent languages, and provides these features for us to use.
- Everything in today's presentation is part of the standard library.

Questions?