# GoLang Concurrency & Parallelism

IMPORTANT Resources

* + <https://go.dev/blog/pipelines>
  + <https://divan.dev/posts/go_concurrency_visualize/>
  + <https://medium.com/hackernoon/dancing-with-go-s-mutexes-92407ae927bf>
  + <https://medium.com/@tilaklodha/concurrency-and-parallelism-in-golang-5333e9a4ba64>
  + <https://levelup.gitconnected.com/how-to-implement-concurrency-and-parallelism-in-go-83c9c453dd2>
  + <https://dave.cheney.net/2014/03/19/channel-axioms>
  + <https://gobyexample.com/non-blocking-channel-operations>
  + <https://go.dev/blog/waza-talk>
  + <https://www.cs.cmu.edu/~crary/819-f09/Hoare78.pdf>
  + [https://go.dev/doc/effective\_go#concurrency](https://go.dev/doc/effective_go" \l "concurrency)
  + <https://betterprogramming.pub/cloud-native-patterns-illustrated-fan-in-and-fan-out-daf77455703c>
  + <https://go.dev/blog/context>

## Topics

* Channels
  + Go routines
  + What is channels?
  + Channel direction
  + Unbuffered and buffered channels
  + Semaphores
  + Pipelines
  + Select statement
* Synchronization
  + Mutex
  + Atomicity
  + Semaphores
  + Channels (for comms)
* Fan-Out Fan-In pattern
* Cancelling go routines
  + Stopping short
  + Explicit cancellation (through channels and Go context package)
* Rate-limited server (good exercise ✅)
* Vector operations (✅)

# FanOut-FanIn Pattern

Fan-out/fan-in refers to the pattern of executing multiple functions concurrently and then performing some aggregation on the results