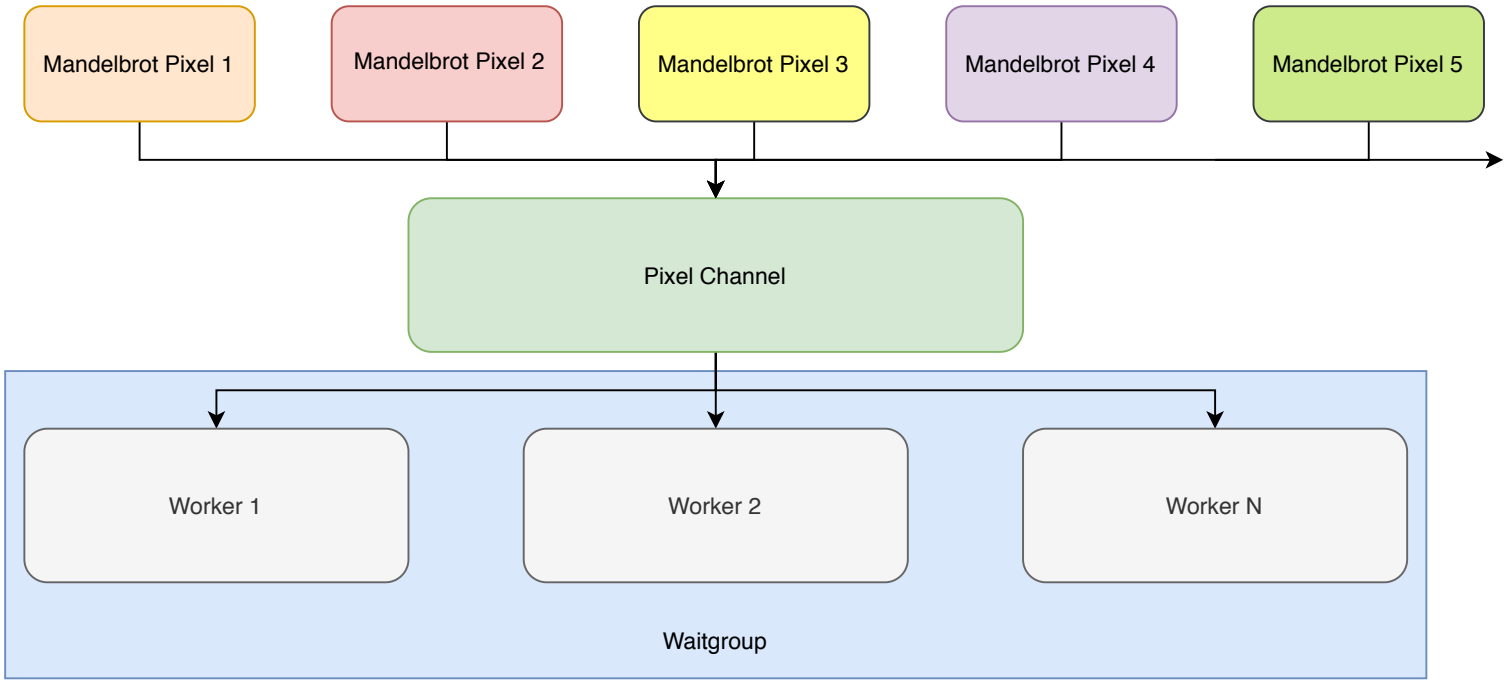
WORKER-POOL PATTERN

In this approach, we do not split the picture into sub-rectangles. We want to maximize the use of goroutines for one "pixel" or more precisely one computation of the mandelbrot algorithm.
However, even if a goroutine is a light-weighted process this could grows in size fast and breaks the stack easily as we have thousands, or even an infinity of goroutines running concurrently considering we can change the resolution.
The solution is to implement a worker pool inspired from the threadpool pattern.

We define a number of workers (here runtime.NumCPU is used)
These workers are fetching constantly from a shared channel in order to set the pixel color into the buffer which.



All mandelbrot computation are made concurrently. Each time one computation is finished, its result is sent through the channel and is received by a worker. if all worker are working, it is added to the "internal" queue of the channel waiting to be processed.