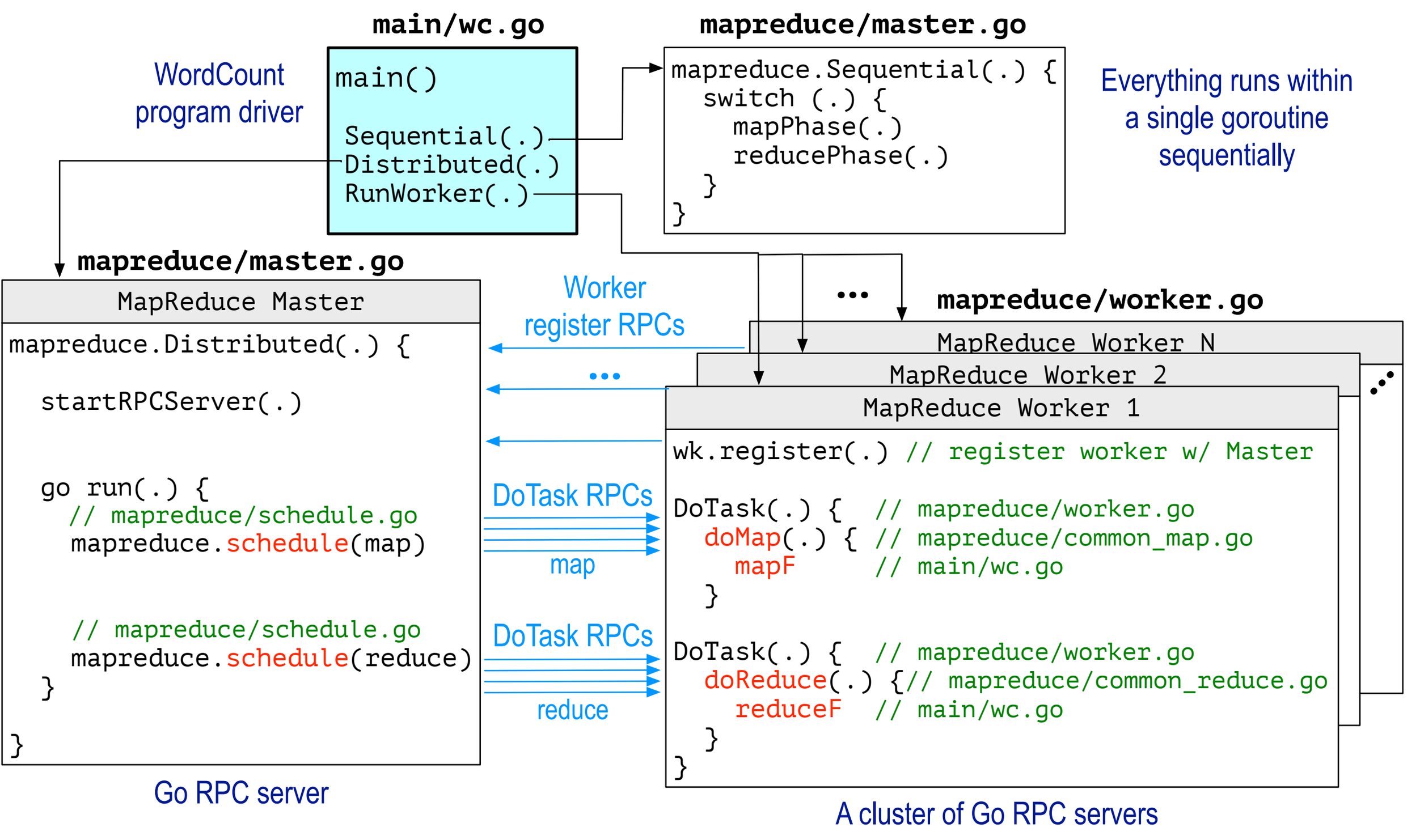
main/wc.go
WordCount program driver
main()
Sequential(.)
Distributed(.)
RunWorker(.)
mapreduce/master.go
mapreduce.Sequential(.) {
switch (.) {
mapPhase(.)
reducePhase(.)
}
}
Everything runs within a single goroutine sequentially
mapreduce/master.go
MapReduce Master
mapreduce.Distributed(.) {
startRPCServer(.)
go run(.) {
// mapreduce/schedule.go
mapreduce.schedule(map)
// mapreduce/schedule.go
mapreduce.schedule(reduce)
}
}
Go RPC server
Worker register RPCs
...
DoTask RPCs
map
DoTask RPCs
reduce
mapreduce/worker.go
MapReduce Worker N
MapReduce Worker 2
MapReduce Worker 1
wk.register(.) // register worker w/ Master
DoTask(.) { // mapreduce/worker.go
doMap(.) { // mapreduce/common_map.go
mapF // main/wc.go
}
DoTask(.) { // mapreduce/worker.go
doReduce(.) {// mapreduce/common_reduce.go
reduceF // main/wc.go
}
}
A cluster of Go RPC servers