DS project documentation

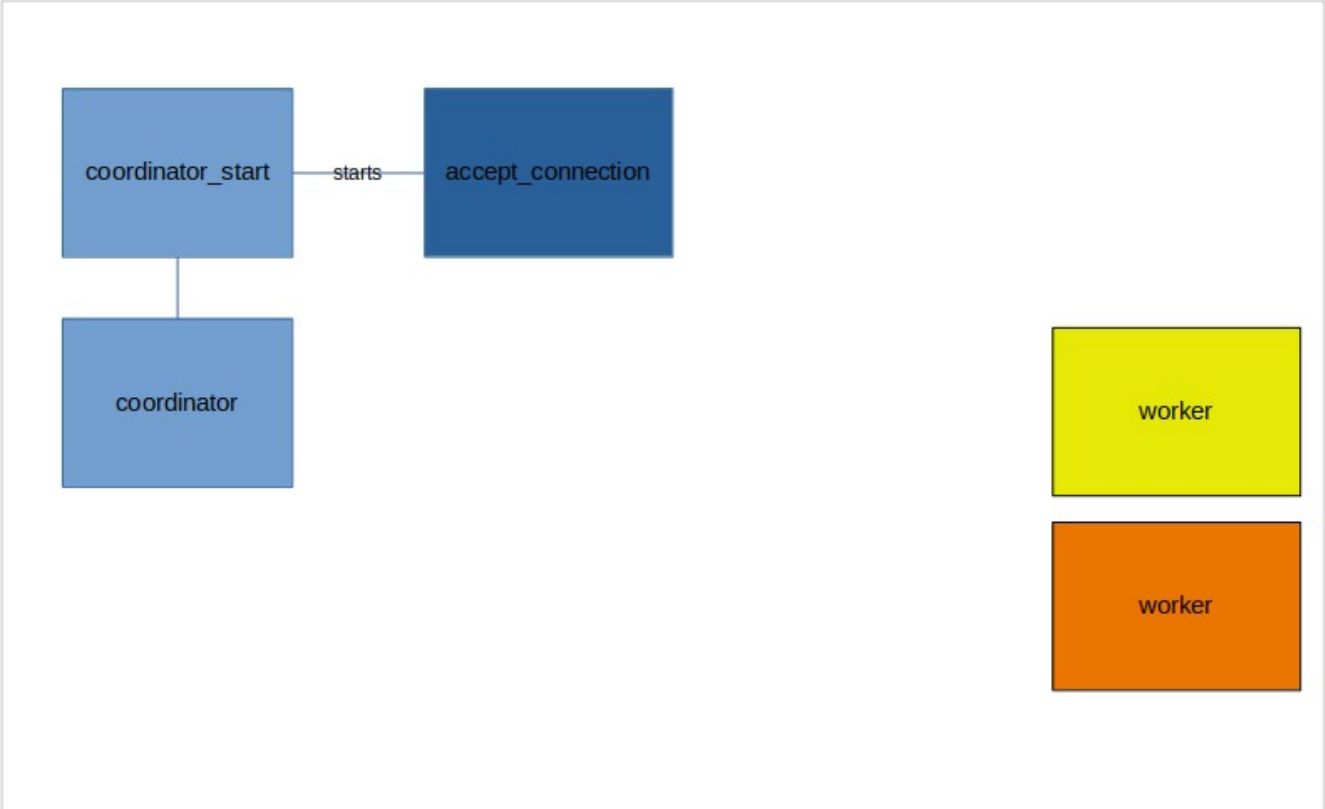
An informal documentation for the module ds.erl

FUNCTIONS

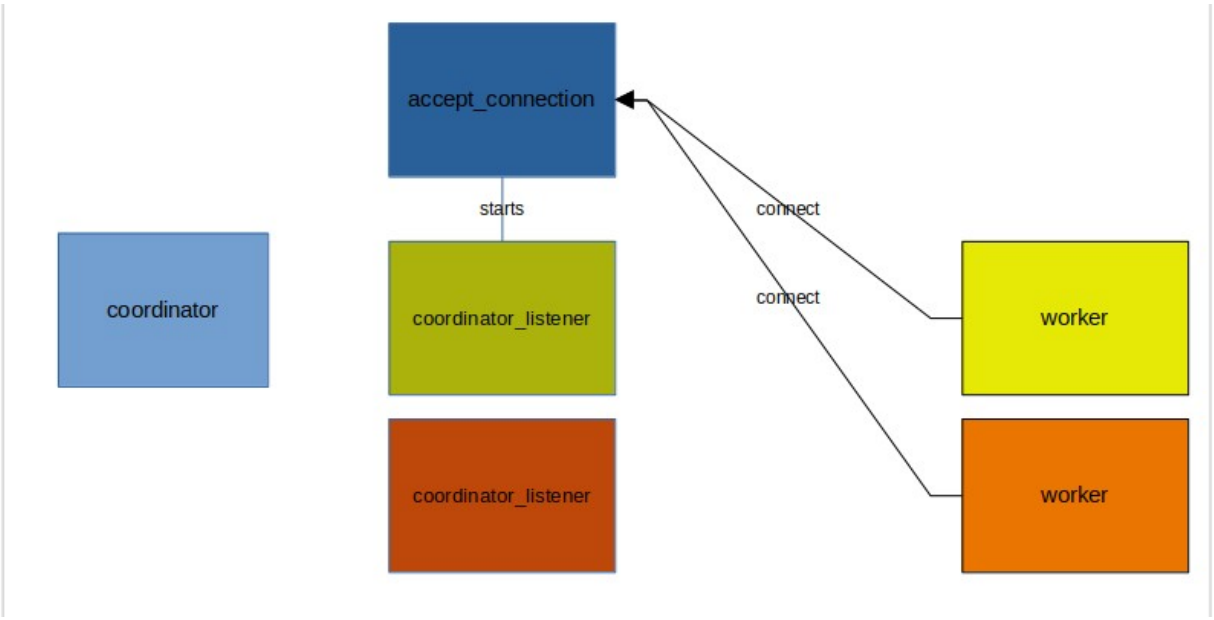
worker/0	worker starts a worker, with no parameters, will call worker/2 with default parameters: “127.0.0.1”, 8080
worker/2 (Host :: atom Port:: atom)	worker starts using host and port parameters. Worker will open a connection to the coordinator and send a “join” message. Then, the worker will wait for a {“work”, function, input_list} message to execute the function over the list of inputs
coordinator_start/1 (Port :: atom)	coordinator starts the coordinator, calling coordinator_start/2 with default settings (Port = 8080)
coordinator_start/1 (Port :: atom)	starts the coordinator. Will launch a pid for accept new connections and will call coordinator/2 for waiting the first worker to join the computation cluster
coordinator/1 (NewReadyW :: [Sock])	coordinator waits for the first worker to join and for the user to decide whenever the coordinator can begin the dispatch of works. When, at least one Worker is connected, and the user agree to start the dataflow program, the coordinator call dispatch_work that starts the dataflow program.
coordinator_start/2 (Port :: atom FileName :: atom)	NOT YET IMPLEMENTED - coordinator starts with port parameter and file name
accept_connections/1 (Port :: atom)	accepts connections and spawn a new Pid as coordinator_listener/1 for each new connection from workers.
coordinator_listener/1 (Sock :: socket)	coordinator_listener waits for the reception of messages by its specific worker, to then set as ready / busy / disconnected the worker to the coordinator. Also, receives the results and forward them to

	the coordinator
dispatch_work/1 (ReadyWorker :: [Sock])	dispatch_work read from the file the input and function and then call two functions send_work/4 and receive_work/2.
send_work/4 (ReadyW :: [Sock] Function :: Input :: [[input]] BusyMap :: map(Sock, [input]))	Send_work, given a list of ready workers' sockets, a function and a list of lists of inputs (the tuples <k, v> or a list of with a map) and a Busy workers'socket – input map, send the next job to the next ready worker and if it cannot send the message, reschedule the remaining inputs to other workers

Startup



Connection



Work assign and result recollection