Parallel:

Build: 接收长度和宽度并生成一个指定长度x宽度的2D矩阵

makeImmutableworld: 将指定的世界转换为函数，转换后只能被读取，不能被修改

findAliveCells: 返回世界中所有存活的细胞

countAliveCells: 返回世界中活细胞数量

calculateNextstate：会计算以startY列开始，endY-1列结束的世界的下一步的状态、

首先将要处理的world部分的数据映射到worldNextState上，计算每个点周围的邻居并将状态写入worldNextState

当死亡的细胞邻居刚好为3个时复活，存活的细胞邻居少于2个或多于3个时死亡。当细胞发生变化时，都要利用CellFlipped 通知 GUI 单个单元格的状态变化。

CountlivingNeighbour： 通过调用 isAlive 函数判断一个节点有多少存活的邻居，返回存活邻居的数量

IsAlive：判断一个节点是否存活，支持超出边界的节点判断（上方超界则判断最后一行，左方超界则判断最后一列，以此类推）

outputPGM： 输出世界，ioOutput通道会每次传递一个值，从世界的左上角到右下角

worker：将任务分配到每个线程

timer：

controller：

s： 生成当前状态的 PGM 文件

q： 生成包含前状态的 PGM 文件，然后终止程序

p：暂停，再次按下时继续（暂停状态下q和s不作用）

distributor： 分工和其它go协程交互

尽量让所有worker分配到的工作相对平均，为避免出现让最后一个worker做完剩余全部的工作，利用变量restHeight来将除不尽的部分分配到靠前顺序的线程上。最后，利用方法makeImmutableWorld将世界转换为函数来防止其被意外修改。

设置ticker，每两秒报告存活细胞数量，同时利用互斥锁来防止竞争。

修改部分：利用go语言特性，在distributor中直接创建两个子线程，避免传入参数。