Tammo Johannes Herbert (319391)
Rico Jasper (319396)
Erik Rudisch (343930)

Distributed Asynchronous Calculation of Conway's Game of Live

Introduction

The Conway's Game of Live is a simple mini universe which is basically a two-dimensional grid of live or dead cells. The progression of time is modeled by ticks where four rules define the transition between two ticks. Following those rules cells become either alive or dead. The goal of this work is to design an algorithm that is able to calculate this mini universe on parallel hardware with high utilization. The result of this calculation is a cuboid which consists of multiple layers representing the progression of time of the two-dimensional cell grid. The first layer is given by an arbitrary initial state of the mini universe.

Definitions

The following terms will be used to explain our ideas leading to our approach:

$c(x, y, t)$ is the state of a cell at the point x, y at the t^{th} tick, which is either live (l) or dead (d).

w is the width of the grid, h is the height. Therefore, $x \in \{1, \dots, w\} = X$ and $y \in \{1, \dots, h\} = Y$. T is the end of time. Consequently, $t \in \{0, \dots, t_{end}\} = T$.

$F(t)$ is a $h \times w$ matrix representing the field $c(X, Y)$ at the t^{th} tick. The successor $F(t+1)$ of the field $F(t)$ is calculated using Conway's rules r use. We call this process a transition $F(t) \overset{r}{\rightarrow} F(t+1)$, which applies the rules r. This will construct the cuboid, which we call the universe U, consisting of all fields $F(T)$

Approach

A simple approach for calculating the Game of Live is the division of the universe U into n little cuboids (subuniverses), each responsible for a subspace of U. We call those U_i (where $i \in \{1, \dots, n\} = I$). Those consist of subfields F_i of contiguous cells. A subuniverse U_i is also a stack of subfields F_i for all ticks $t \in T$. Each subuniverse U_i can be assigned to a node, which develops each layer (field). Unfortunately, a field $F_i(t)$ may depend on another field $F_j(t-1)$ which requires message exchange between nodes.

The simple approach would be that every node calculates the same tick of its respective subuniverse U_i. Thus, this is a synchronized approach. This way the entire field $F(t)$ is calculated. When every node is done, the next tick is calculated. Since some subfields can be calculated faster than others it could cause idling nodes.

The progression of the field development is depicted in Figure 1. For simplicity, only one space dimension is shown. The other dimension is the timeline of the ticks. Colored cells have already been calculated. Note that cells depend on neighboring cells from the layer underneath.

Tammo Johannes Herbert (319391)
Rico Jasper (319396)
Erik Rudisch (343930)

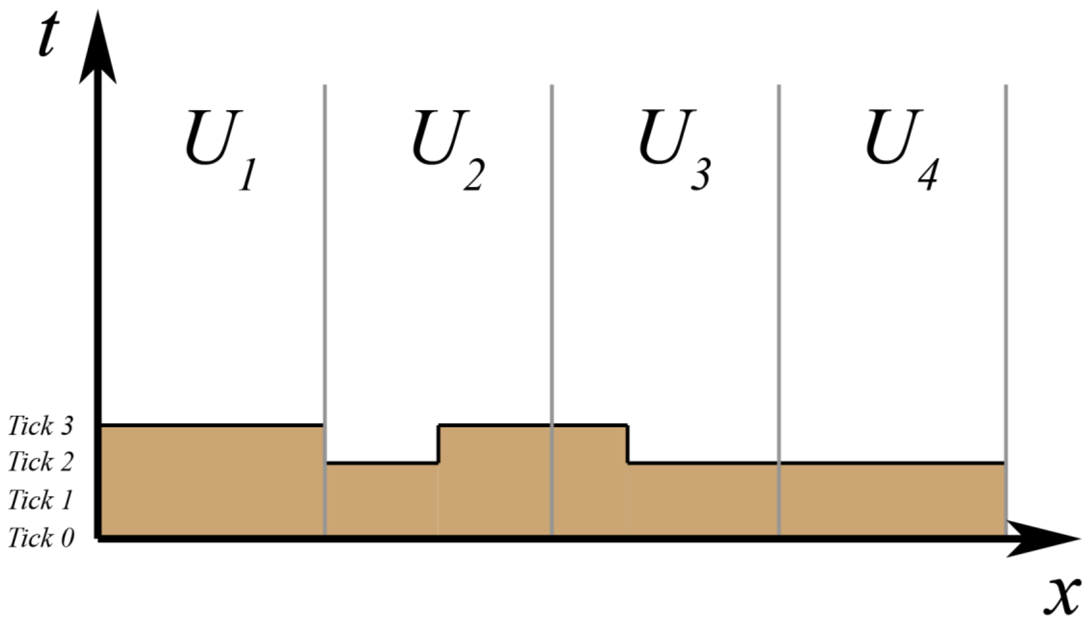

Figure 1: Synchronous tick calculation

One strategy to avoid idle time is to calculate border cells before core cells. Therefore those can be messaged to other nodes which rely on this data. If a node cannot calculate its border because the necessary data is not yet available it can calculate core cells.

Based on this idea the progression of the calculation could be like in Figure 2. Here, the nodes are not necessarily on the same level. A node doesn't have to idle only because needed border material isn't available.

Tammo Johannes Herbert (319391)
Rico Jasper (319396)
Erik Rudisch (343930)

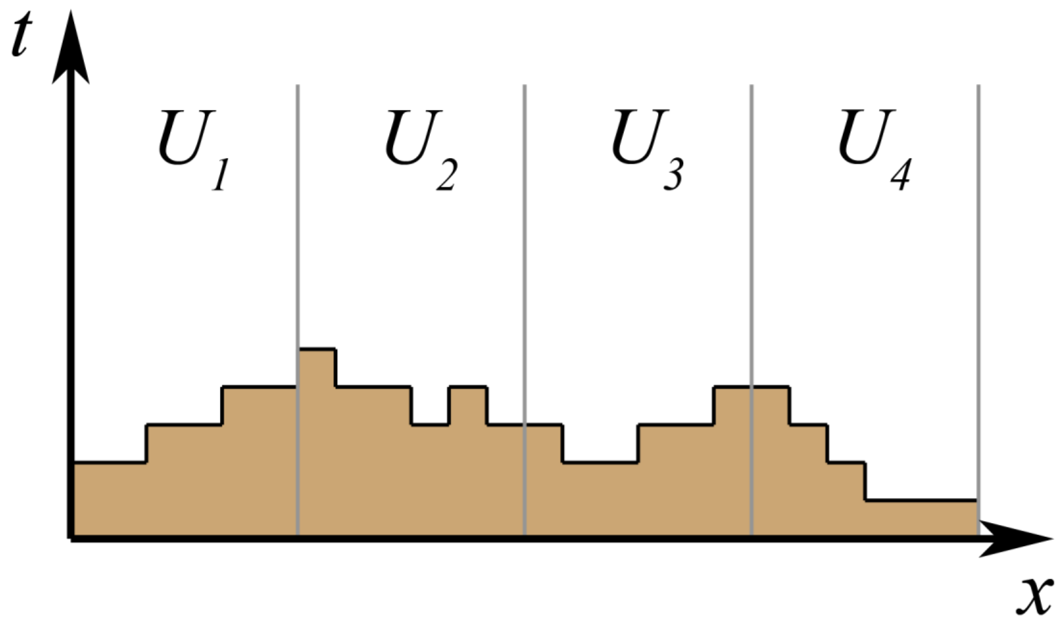

Figure 2: Asynchronous tick calculation

We propose to calculate always the border if possible otherwise cells near the border. This way, after the border of an adjacent subuniverse is available again, the node can calculate its own border as fast as possible.

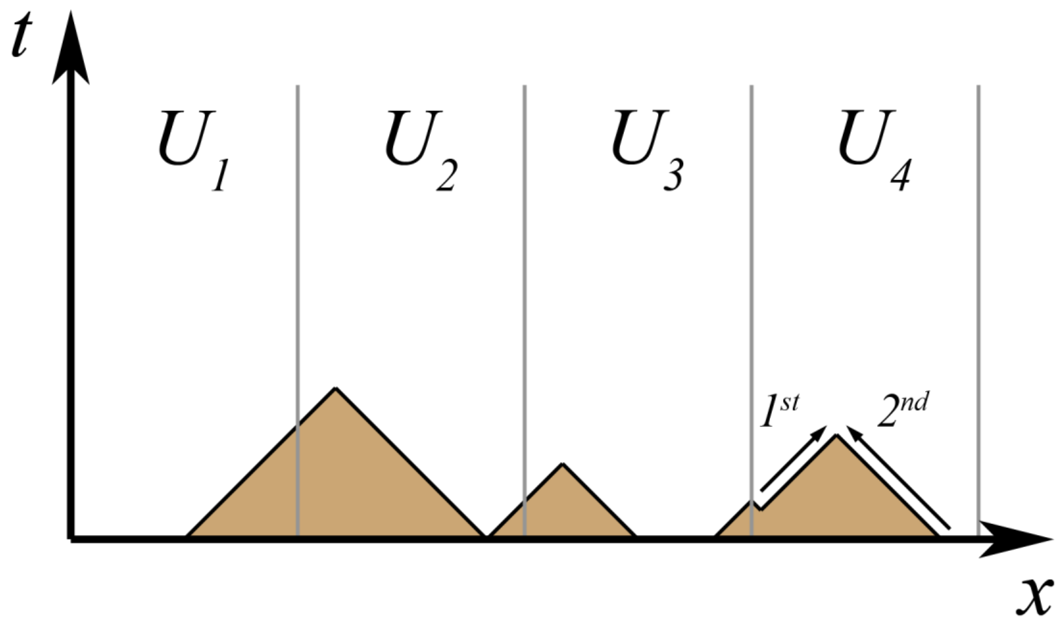

Figure 3: Strategy - first border then core

Tammo Johannes Herbert (319391)
Rico Jasper (319396)
Erik Rudisch (343930)

However, the transmission of border data could still be a bottleneck. Rather than just transmitting a cell-thick border the nodes could also share a border width a width of multiple cells. Another solution is that some threads are purely destined for calculating the border between ordinary subuniverses, depicted in Figure 4. For instance U_1 and U_2 have calculated the brownish area. Now they have to wait constantly for each other. Another thread could calculate the orange area based on the yellow data while U_1's and U_2's cores are calculated. After the orange area has been calculated U_1 and U_2 can proceed calculating the border areas.

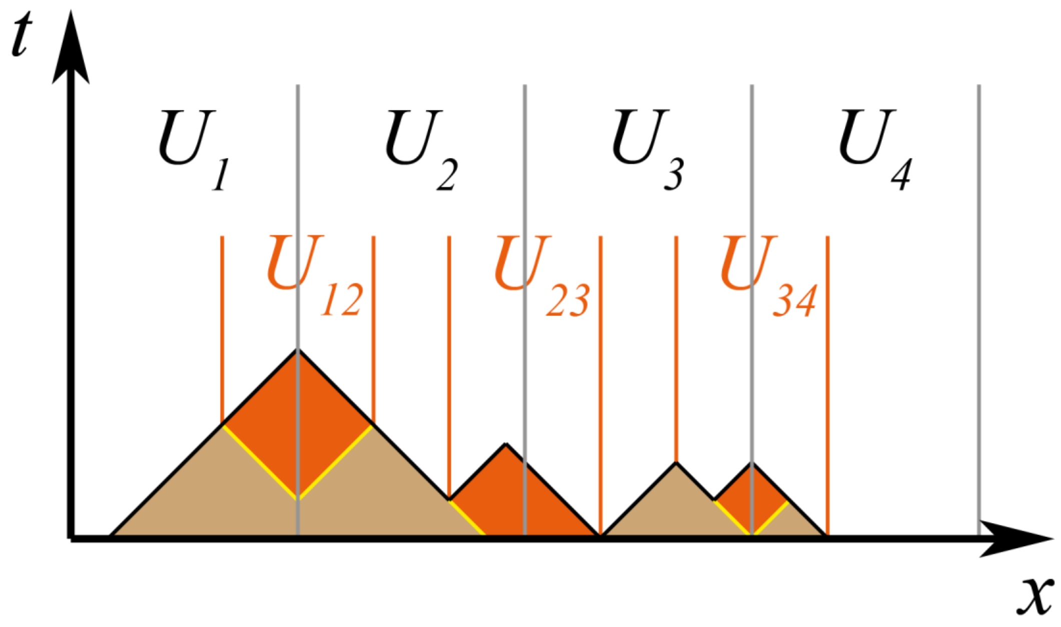

Figure 4: Intermediate universes