**Umeå University**
Department of Computing Science

# Parallel Programming 7.5 p
# 5DV152

## Final Programming Assignment
## MPI Game of Life

Submitted 2017-03-22
Author: Lorenz Gerber (dv15lgr@cs.umu.se lozger03@student.umu.se)
Instructor: Lars Karlsson / Mikael Ränner

## Contents

## 1 Introduction

Here it was the aim to design and implement conway's game of life that can be run on a High Performance Computing system using the MPI library.

## 2 Conway's Game of Life

Conway simplified mathematical models from von Neuman that were about structures that could build copies of themselves. Published 1970, the surprising property of the program was that it represented a turing complete system despite using only very simple rules. It was the start of a new research field about 'cellular automata'.

Conway's game of life is represented by a regular grid of limited size consisting of individual cells that can either be death or alive. The game of life proceeds in rounds: In each round the state of each cell is evaluated individually and then updated to an eventual new state. Then the next round starts. The most important feature of this system are the rules applied for assessing and updating the state of each individual cell: Based on the eight neighbouring cells, it is determined if a cells environment is life promoting or overcrowded. Hence, the variable to determine and evaluate is the number of neighbouring cells alive.

## 3 Design and Implementation

From the description of the game itself, some preliminary design drafts were sketched. Then a short litterature study about available implementations was conducted. Several programs of interest were found and inspected for inspiration [1] [2] [3]. To be flexible, it was decided to implement both randomized game start state and loading of a saved state from file. Because graphics programming was not the focus of this course a simple script in R [5] was written for visual inspection of the game state after a number of generations. 'Foster's Methodology' as reproduced in the course litterature [4, p. 66] was applied.

1. Partitioning
   Two task were identified: Detecting the number of neighbours alive and updating the state for the next round. However, as it will be seen later, in the current case, it could make more sense to see each single cell as an individual task.

2. Communication
   The two first mentioned tasks above need to run in sequence and they also need to be concerted among the whole game. If a single cell is seen as an individual task, then the communication would be the step where the neighbouring cells alive are determined. The cell is locally dependent to access the eight surrounding cells.

3. Agglomeration / Aggregation
   Agglomerating the individual tasks per cell seems to make sense as they adhere to a strict temporal run sequence. However more interesting is to consider agglomerating larger numbers of individual cells. As most of the cells would have eight neighbours within the group, they could be considered to be independent with a few special cases for the cells on the interface/border.

4. Mapping
   As the identified tasks are tightly coupled, it seems to make most sense to map larger

agglomerates of cells to individual MPI nodes. Hence, communication during the game will be limited to handling the mentioned special cases of cells that are at the border of a local cell group.

Practically, it was decided to divide the global square game of life field vertically into equal sized local rectangular fields of cells. As such, communication between nodes includes only the cells that are on the vertical borders as the horizontal borders (top and bottom) can be calculated locally. To facilitate the communication, the local rectangular cell groups are padded with one layer of cells where the cell state from other local border interface can be imported to.

Hence there were a total of three tasks per game cycle determined:

1. update interface region
2. determine number of neighbours alive
3. update state

## 4 Usage Instructions

### 4.1 Building

### 4.2 Usage

**Running the Program**

**Graph Output using R Script**

## 5 Validation

Here the qualitative result of the serial and parallel version were evaluated to yield the same result. This was done using a batch script that runs both programs and then compares the output files using the system tool diff.

## 6 Benchmarking

## 7 Known Issues

## References

[1] Game of life c serial. `https://rosettacode.org/wiki/Conway%27a_Game_of_Life`, 2017. accessed: 2017-03-23.

[2] Game of life mpi, kth summer school. `http://www.pdc.kth.se/education/summer-school/mpi-exercises/mpi-lab-codes`, 2016. accessed: 2017-03-23.

[3] Game of life with mpi. `http://www.shodor.org/petascale/materials/distributedMemory/code/Life_mpi/`, 2002. accessed: 2017-03-23.

[4] P.S. Pacheco. *An Introduction to Parallel Programming*. Morgan Kaufman, 2011.

[5] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

## A C Source Code TSP Condways Game of Life