1 Approach

Our group developed an MPI-only solution that, conceptually, creates a ring of nodes in which every nodes communicates with its two adjacent neighbours at the start of every iteration to exchange their shared boundary values.
Morevover, we sought to distribute memory allocation as evenly as we could between each process in order to increase the range of solvable problems.

2 Decomposition

Our group used a layer decomposition of the problem, which corresponds to $2 \times \frac{N \times N}{P}$ points of intersection in total between problems. We are aware of better decompositions, however the limited manpower available proved fatal in the implementation of a checkered decomposition which would provide a reduction of up to 75% in the intersection between process' problem domains.

3 Concerns

In parallelizing the computation of Game of Life 3D our group's biggest concerns were:

1. Avoiding deadlocks between processes

Namely, eliminating possible situations in which two processes lay in wait for sending or receiving data from one another.

2. Efficient exchange between processes

Avoiding waiting chains in which process n sends data to $n + 1$ which only upon receiving such data sends its data to $n + 2$ which is already waiting for it, and so on.

3. Load balancing

Minimizing the maximum difference between any two processes work load, as each iteration can only be as fast as the slowest of the processes. Our approach to this problem is described in section 4.

4. Problem domain intersection

As described in the above section, we sought to minimize the intersection between each process' problem domain to its neighbouring processes' problems to reduce the ratio between the time spent effectively calculating the solution and the time spent transferring data between processes, i.e., the overhead introduced by the problem's decomposition.

4 Load Balancing

The problem, considering cube of size N and P processes is divided in layers of size $\lfloor \frac{N}{P} \rfloor + Nmodp$, such that the disparity in layer size between every two processes is minimized to a maximum of 1.
Furthermore, the memory allocation and initial grid generation are each owner processes' responsibility. The seed is sent from process n to process $n+1$ after the former completes its grid generation, with the intent of reducing the amount of memory allocated by the master process.

5 Results

5.1 Sample Results

Program Parameters				Number of Processes						
g	n	ρ	seed	1	2	4	8	16	32	64
1000	64	0.4	0	25.0	17.2	8.9	4.7	3.0	2.0	1.1
200	128	0.5	1000	51.7	29.4	14.0	7.0	3.9	2.1	1.3
10	512	0.4	0	174.7	116.4	60.2	30.9	16.7	9.2	5.0
3	1024	0.4	100	436.9	295.7	145.0	83.9	47.7	29.6	20.7

Table 1: Sample Results (in seconds)

5.2 Result Discussion

Our results show an efficiency of ~ 75% for 2 processes that incrementally decreases to ~33-35% due to an increase in the overhead in communication between processes due to the increase in intersection of problem domains, as expected.

Note

This project was developed by Alexandre Coelho, student number 100120, alone.