1. Introduction

This exercise is comprised from two parts. In the first part, we see how to manage shared memory among processes.

In the second part, a load balancing algorithm is to be implemented. Should the parallel processes have a really unbalanced load among each other, a load balancing task should be performed. Assume you have \( N^2 \) processes in some arrangement. In our cases, one of them gets all the tasks at the beginning and wishes to divide its load among others.

2. Shared memory – the code

The following code creates two threads, which each increment a shared integer. Discuss what happens, when the lines `#pragma omp critical` are not included – why the value of the shared integer is not as it should be.

```c
#include <stdio.h>
#include "omp.h"

int main(int argc, char** argv) {
    long commonInt = 0, REPEAT = 1000000;

    #pragma omp parallel sections shared(commonInt)
    {
        #pragma omp section
        {
            for (int i=0; i<REPEAT; i++)
            {
                #pragma omp critical
                { commonInt++; }
            }
        }
        #pragma omp section
        {
            for (int j=0; j<REPEAT; j++)
            {
                #pragma omp critical
                { commonInt++; }
            }
        }
    }
    printf("End value: %ld\n", commonInt);
    return 0;
}
```

3. Load balancing task

Your job is to implement the diffusion model (presentation slides 3-68 and 3-69) in a program and language of your choice (for example MatLab). The arrangements you have to consider are:

- a N times N grid, where most of the elements have 4 neighbours – see Figure 1,
- a circle with N² elements, where all elements have 2 neighbours, and
- a N times N torus, where all elements have 4 neighbours.

For start, N=4 is a good value. Conditions which should be met are:

- the number of tasks assigned to an individual process is a nonnegative whole number;
- all the processes expect for one start without any tasks;
- none of the tasks must get lost (i.e. the sum of the tasks on processes at any time is equal to the starting number of tasks).

The model, upon which the load should be swapped between processes, is:

$$w_i(t+1) = w_i(t) - \sum_{j \in N(i)} \alpha \cdot (w_i(t) - w_j(t)), \quad 1 \leq i \leq N, -1 < \alpha < 1$$

where \(w_i(t)\) is the workload done by process \(P_i\) at time \(t\). Find an appropriate value (range) for \(\alpha\).

![Figure 1: 4 times 4 grid with initial values and the values of first two steps. \(\alpha = 0.25\).](image)

4. Contact

Should any difficulties arise, please do not hesitate and contact me per email. But remember – Google is your friend (http://www.giyf.com)!

Tutor: Stefan Markic  stefan.markic@tum.de