INFO-F-404 : Operating Systems

2018 – 2019 Project 2 : Mastermind solver

1 Main goal

Implement a parallel *Mastermind solver* using MPI on HYDRA.

2 Introduction

Confronted to the impossibility to persevere the pursuit of Moore's law, the response of the electronic industry to provide always more computing power resides in parallel and multicore/multiprocessor architectures.

In this project, we will be taking advantage of this many cores paradigm by implementing a parallel solver.

3 Mastermind rules

Mastermind is a 2-player game. The first player — the Challenger, must guess a secret ordered list of colours. This secret is called the solution. It is chosen by the second player — the Game Master. Each turn, the Challenger proposes a Guess to the Game Master. The Game Master evaluates the Guess and gives two figures: the number of colours that are in the same spot in the solution — called Perfect, and the number of colours that are in the solution, but not in the right spot — called Colour Only. The game ends when the solution is found.

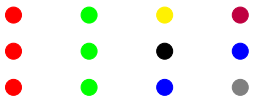
Assuming that the solution contains 4 colours, chosen among Red, Blue, Green, Yellow, Dark, Gray and Purple, and that the solution is the following (Red, Blue, Green, Yellow):

The following guess (Red, Purple, Blue, Green):

will be evaluate as containing one Perfect (Red), and two Colour Only (Blue and Green).

Plausible: *a Guess is plausible iff it does not contradict any previous guess's evaluation.*

For an example, if the first Guess (Red, Green, Yellow, Purple) has been rated with two Perfect, the second Guess (Red, Green, Black, Blue) has been rated with two Perfect and a Colour Only: the last guess (Red, Green, Blue, Gray) is *plausible*:


ColourOnly $= 0$, Perfect $= 2$

ColourOnly $= 1$, Perfect $= 2$

The following guess (Red, Green, Blue, Black):

is not plausible because it contradicts the second guess's evaluation. The four colours are the same, when the evaluation indicates that only three colours from the guess are actually in the solution.

There are two constraints when picking the solution:

- No duplicate are allowed. The following Guess is not possible because it contains two Red:

- Exactly 4 colours chosen. The following Guess is not possible because the second spot is left empty:

4 Project details

You are asked to implement a *Mastermind* solver using OpenMPI on the ULB's Hydra cluster.

We can assume that the solution has four spots and that there are ten different colours. However, your solution must be generic. We must be able to change the number of colours and spots easily.

The solver will be composed of two components. The Game Master will be played by a simple component on a single master node. This component must choose a random possible solution, and to evaluate each guess.

The Challenger will be implemented on a distributed component, on several nodes. Each node may propose a *plausible* guess to the master node each turn.

The code must be modular, with a single class per file. The clarity of codes (and comment) will be evaluated.

5 Algorithm details

The master node must first choose a solution before the game starts. Then, it must receive each turn a guess from each computing nodes, pick one plausible and evaluates it. It then prints the guess and its evaluation. Finally, it sends the guess with the evaluation to the computing nodes for future turns.

The computing nodes must compute all the possible guesses using brute force at first. This computation must be distributed between all the nodes: each node must store only a part of the possible guesses. For an example, if there are four colours and four computing nodes: the first stores the guesses that starts with the first colour, and so on. Each turn, each computing node tries to find a plausible solutions. It compares the list of possible guesses to the evaluations received previously and send the first plausible to the master. If a computing nodes has no plausible guess, it sends an error code to the master slave. The first proposed guess is either arbitrary or random.

5.1 Report

You should also write a short report that contains:

1. short description of your code (diagrams) and implementation choices,
2. a description of the protocol that you used to implement the program (messages transiting between processes),
3. a description of the performances (and limitations) of your implementation,
4. a formula giving the number of possible guesses at start depending on the number of spot and colours,
5. a section where you describe difficulties that you met during this project (and solutions that you found).

6 Submission and planning

This project should be done in groups of 3 maximum, you may choose your partner(s).
It has to be submitted before 23:55:00 o'clock on December 14th, 2018.
Your project has to work properly (compile and execute) on HYDRA cluster at ULB.

You have to submit (in a *zip* file) a folder that contains at least the following files:

- your C++ source, with a Makefile,
- a short report (pdf format, figures and graphics are welcome).

The zip file with your project has to be submitted to the UV: `http://uv.ulb.ac.be`. The name of the folder and of the zip file will be as follows: if Jean Dupont made his project with Albertine Vanderbeken, they should send a file named *dupont-vanderbeken.zip* (that contains a folder of the same name, that contains all your project files).

For questions on this project, please refer to **xavier.poczekajlo@ulb.ac.be** with subject "[INFO-F-404] project2".

Good work!