CS3210: Parallel Computing – AY 2013/2014 Semester I

Assignment 2: MPI Basketball
Individual Submission (20 marks)

Deadline: **10 Nov 2013**

Distributed-memory organization allows for performance to be scaled in parallel computing. The learning objective of this assignment is to reinforce your learning of message-passing programming by simulating a basketball game. This assignment is divided into three parts: (1) **basketball training** session using collective communication, (2) **basketball match**, and (3) **performance challenge** to reduce the parallel computing time for simulating the basketball match (*best three submissions get bonus marks*). The game must be programmed using MPI. Each player must be implemented as a separate MPI process and the field must also be implemented as MPI process.

The basketball court is 128 by 64 feet. You must model the field as a grid divided into square grid element of one foot. Players move from one grid element to another in increments of one foot. The player can move in four directions: top, right, bottom or left. There are two basket posts on the court, located on the extreme ends, at coordinates (0, 32) and at coordinates (128, 32). The ball is initially at the center of the court (64, 32). It is your strategy to determine the starting locations of the players. Basketball players cannot be trusted with too much information otherwise they easily lose focus. Therefore, a player works as follows: whenever he needs to find out the location of the ball or the location of another player, he must consult the field process for this information. **The player is not allowed to guess or to know beforehand the location of the ball or of another player.** This is a strict requirement in this assignment. However, each player knows its own location.

A. Basketball Training Session – use collective communication only (8 marks)

Write an MPI program to simulate a basketball training session. The training consists of only one team, and the players of the team pass the ball around to warm-up and practice their shots for the upcoming match ☺.

Your program must be structured as follows:

- There must be 5 player processes. Each player processes only know information about itself. If a player requires any information about other players or about the ball, the player must obtain it from the field process.
- There is one field process that disseminate to player processes information about the location of the ball or of other players.
- **For the basketball training session, use only collective communication. You are not allowed to use any type of point-to-point communication.**

Each training session consists of 900 rounds. One round proceeds as follows:

1. All players determine the grid element where the ball is located. The ball is assumed to be stationary.
2. All players start running towards the ball from a distance, randomly between 1 and 10 feet:
   A. If, within their distance, they could reach the grid element with the ball:
      1) If they are alone on the grid, they win the ball.

2) Else, one of the players randomly wins the ball. (You are free to use any mechanism to determine the winner, provided that (1) only one winner is selected, and (2) the winner is selected randomly).
3) The winner of the ball passes the ball to a random location on the grid. The winner informs the field process of the new ball location. (The location of the ball changes (at most) one time per round)

B. If they do not reach the ball, they stop after running and remain on that grid element.

3. The players send their new location to the process of the field in charge of their location.

Each player must collect the following pieces of information about his own training:
1. How many feet he has run.
2. How many times he has reached the ball.
3. How many times he has passed the ball.

After each round, the field process must output to the console the following information:
1. Number of the training round (starting from 0)
2. The new coordinates of the ball, with dimension on long side of the field first.
3. For each player, starting from id 0 to id 4: Id, initial coordinates, final coordinates, whether they reached the ball (0 or 1), whether they pass the ball (0 or 1), feet run so far, number of times he reached the ball and number of times he passed the ball.

Example:

…
15 // This is round 15
119 20 // Ball is at coordinates (119, 20). Coordinates are given as long side of court first.
0 106 21 109 25 1 0 78 2 1 // Player 0: initial location (106, 21) final location (109, 25), 1 = he has reached the ball this round, 0 = he did not win the ball this round, 78 = he ran 78 feet from the beginning of the training, 2 = he reached the ball twice since the beginning of the training, 1 = he won the ball once from the beginning of the training.

B. Basketball Match – use either collective communication or point-to-point communication or both (12 marks)

Write an MPI program to simulate a basketball match between two teams.

Field (Court)
For the basketball match, you must simulate two teams of players (Team A and Team B) and one field. This time, the field consists of 2 processes, where each process covers one half of the court. Field process 0 (FP0) will be in charge of the part between corner (0, 0) and corner (64, 64), while field process 1 will be in charge of the part between (64, 0) to (128, 64). It is up to your implementation to assign the middle line to either FP0 or FP1 (or both). The assignment of field grid elements to MPI processes is shown in Figure 1. You must respect this mapping in your implementation. Note that the field processes are in charge of the locations, not the players.

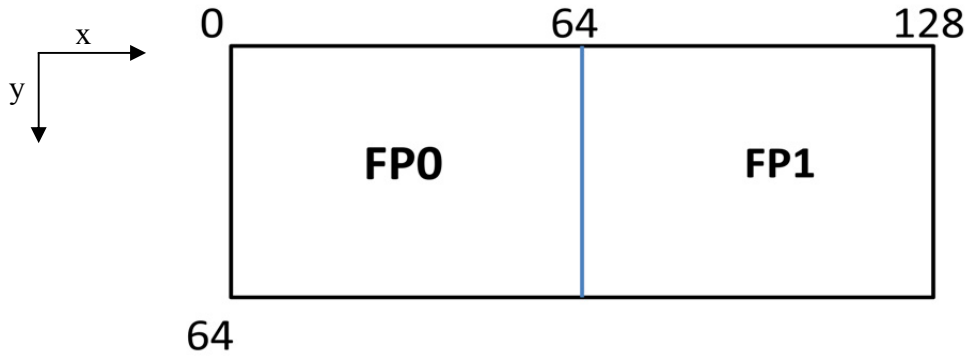

Figure 1: Assignment of field locations to MPI field processes

Players

The players have a more complex behavior compared with the training session. Each player has three attributes:

1. **Speed** - dictates the maximum distance the player can run in one round.
2. **Dribbling skill** - influences the likelihood that the player wins a challenge for the ball or the likelihood that he keeps the ball, if challenged by another player.
3. **Shooting skill** - dictates the probability the player will shoot the ball in the desired location.

All attributes have values between 1 and 10. For fairness, each player has speed + dribbling + kick = 15. That means a player can be very good at one attribute and bad at the others, or mediocre at all three.

Game

A basketball game is made up two halves, each consisting of 2700 rounds. In the first half, Team A guards the left post and attempts to score at the right post, while Team B guards the right post and attempts to score at the left post. In the second half it is the other way around.

A round proceeds as follows:

1. Players determine the grid element that holds the ball (by asking the corresponding field process that is in charge of their location). The ball is assumed to be stationary.
2. Players start running towards the ball for a **distance**, at most **speed** feet (for each round, it is up to your strategy to decide any **distance** from 0 to **speed**). **You may choose if all or only some players run towards the ball**.
   A. If, within their **distance**, they reach the grid element where the ball is:
      1) If they are alone on the grid element, they win the ball.
      2) Else:
         i. They pick a random number between 1 and 10.
         ii. They multiply this number with the **dribbling skill**. We call this product **the ball challenge.**
         iii. They send **the ball challenge** to the field process on which the ball is located.
         iv. The field process on which the ball is located receives **the ball challenges** from all processes challenging the ball. The field process chooses the winner as the player with the highest **ball challenge.** If two players have the same **ball challenge**, the field process randomly chooses one of them.

3) The winner of the ball determines a location on the field where to shoot the ball. This location can be anywhere in the court. The winner shoots the ball to that target location. The ball may or may not reach the target, depending on the shooting skill and on the distance between the player and the target location. If the ball reaches the location, it stays there until the end of the round. If it misses the location, the field process (which is in charge of the target location) randomly chooses a location within 8 feet of the target location, and the ball is considered to land there.
4) They remain on the grid element.

B. If they do not reach the ball within their **distance**, they stop after running **distance** feet and remain on that grid element.

3. The players send their new location to the field process that is in charge of their location.

The formula for the probability of shooting the ball to a target is:

$$p(d,s) = \min(100, \frac{10+90s}{0.5d\sqrt{d}-0.5})$$

where *p(d,s)* is the probability of reaching the target, expressed as a number between 0 and 100, *d* is the distance between the player and the target, and *s* is the shooting skill.

The distance *d* is between points (x_1,y_1) and (x_2, y_2) is computed as:

$$d = abs(x_1 - x_2) + abs(y_1 - y_2)$$

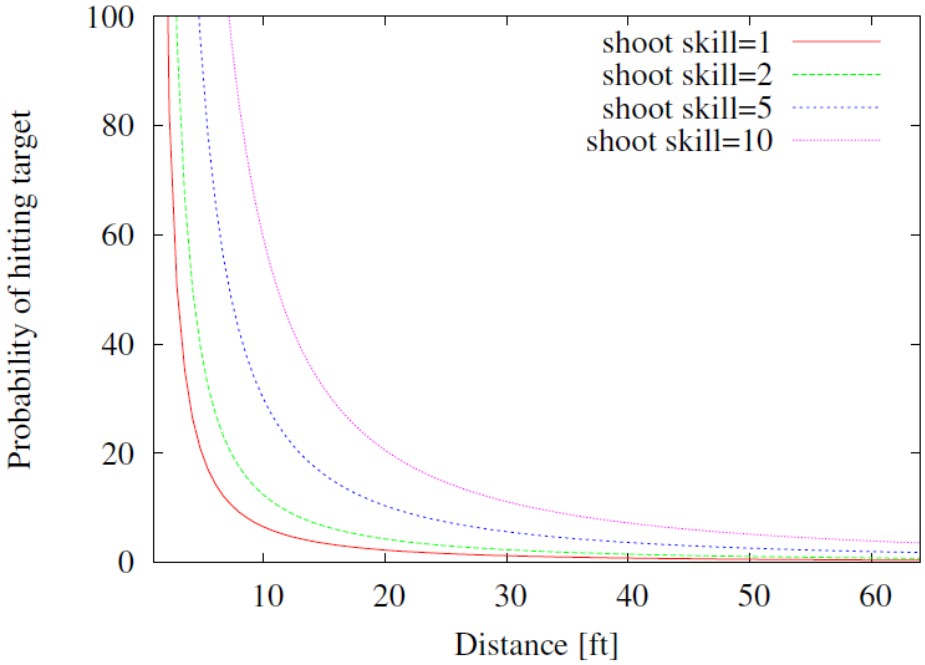

Figure 2: Probability of reaching target as a factor of distance and shooting skill

Figure 2 plots the probability of a player shooting the ball to target, as a factor of distance and shooting skill. As can be seen, a low shooting skill means that the player has a high chance to miss even at distances less than 10 feet. In contrast, a player with shoot skill 10 has almost

100% chances of hitting the target at distances under 10 feet. The probability of hitting the target drops rapidly when the distance increases, and the shoot skill becomes almost irrelevant for large distances. A good strategy is to avoid shooting from large distances, as it is unlikely that the ball will reach the target.

Special cases:

1. If the ball is shot at (0, 32) or at (128, 32) the team has scored a goal. If the player that shot the ball was at a distance of more than 24 feet, the goal is worth 3 points. If the player is at less or equal than 24 feet, the goal is worth 2 points.
2. After scoring a goal, the game is restarted with the ball at the center of the court (64, 32).
3. If the ball is shot out of the court, the game precedes with the ball the center of the court (64, 32).

You are required to implement the following features:

MPI Requirements (7 marks)

1. All communication for the basketball match can be done using either collective communication or point-to-point communications (or both). You are free to choose blocking or non-blocking communications.
2. You must have exactly 12 processes (10 players + 2 field processes).
3. Each team is part of a communicator. You are free to communicate anything between players of the same team, with the exception of the locations of the ball or of other teammates. A player sends its location only to a field process.
4. Players from different teams must not exchange ANY data.
5. Field processes can communicate with each other. Field processes can only communicate with players whose locations they are in charge of.
6. After each round, field process 0 must output to the console the following:
   a. Number of the round (starting from 0)
   b. Score: Team A points, Team B points
   c. The new coordinates of the ball, with dimension on long side of the field first. If a goal was scored in the round, the new coordinates are at the center of the court.
   d. For each player, starting from id 0 to id 4 of team A: Id, initial coordinates, final coordinates, whether they reached the ball (0 or 1), whether they kicked the ball (0 or 1), ball challenge (the challenge is -1 if they have not reached the ball), shoot target location (location is (-1,-1) if they don't reach or don't win the ball). After team A, same for team B.
7. You are free to chose the player attributes, subject to the restriction speed + dribbling + kick = 15, with each attribute between 1 and 10. You can choose the player starting location in the match. The initial starting location of the ball is the center of the court at (64, 32).

Competitive Requirement (5 marks)

8. You must program both teams to be competitive – i.e. an attacking team should attempt to score goals and a defending team should try and prevent the other team from scoring. **Passing the ball randomly does not count as competitiveness.** You can also program Team A and Team B to play by different strategies. You have a lot of processing power that you can harness – be creative ☺

Performance Requirement (Bonus 3 marks)

9. You must measure the time required to execute the 5400 rounds and report the average time required to execute one round. You should attempt to minimize this time. The student that obtains the smallest average time per round is awarded 3 bonus marks. The student with the second best average time receives 2 bonus marks, and the student with the third best average time receives 1 bonus marks. You may comment-out all printing for the runs on which you collect your execution time.

Guideline for Marking

1. Be creative. The more creative strategies and design decisions you have, the better.

2. Be correct. Please make sure that your programs are correct. **You will lose half of the marks for the question whose corresponding program cannot be properly compiled or cannot be properly run (e.g. due to segmentation faults or deadlocks).**

3. Be efficient. Your task is not only to implement the training section correctly but also to implement it efficiently. Avoid misuse or overuse of MPI communication operations and optimize your programs to reduce the communication overhead, i.e. reduce the data exchanged and the number of communication operations (e.g. the collective or point-to-point operations). This is important as the main purpose of parallel computing is to achieve performance.

4. Be clear. The report should be clearly written to clarify your strategies and design decisions, as described in the next section.

   Submissions that satisfy the guideline better will get higher marks.

Requirements for Implementation and Deliverables

Your program **must run on the cluster formed by a workbench in the lab**. You need to upload to IVLE Folder "Assignment 2", under "Student Submission" directory a **zip archive** containing the source code and **a report in PDF format**. The report must contain:

1. Your name and matriculation number.
2. Pseudo-code of your solution for both parts.
3. Diagrams of your communication between the MPI processes, within a round.
4. A walkthrough of your design, highlighting the important design decisions, as well as the skills chosen for each player. State your assumptions with justifications (e.g. the starting locations of the players).
5. A makefile, or instructions on how to build and run your program, as well as the configuration (machinefile or rankfile) which may be needed to run the program.
6. The average time per round.

In addition to the report, the zip should contain a sample of the output of each program (you need to redirect the output to a file; add this file to the zip).

You need to provide two source code files, one for the training session and one for the basketball match. The source code must be properly commented (4 marks penalty) and indented (2 marks penalty).

Please name your submitted ZIP file as follows:

Nusnetid-assignment-2.zip

There is a penalty of 1 mark for each of the above requirements if it is not met.

Deadline is 10 November 2013, 23:59. Penalty for missing the deadline is 2 marks per day.