Implementing Connect4 game using Scala Parallel features

Authors:

1. Naga Venkata Sai Indubhaskar Jupudi
2. Trivikram Bollempalli

Introduction

- Two player game in which each player chooses a color and take turns to drop a coin in a vertically suspended grid.
- One who places his coins in contiguous 4 locations in any specific direction, wins.
- Scala, a Functional programming language designed to be a superior version of Java.
- Parallel programming primitives of Scala will improve the performance of the application by leveraging the underlying CPU resources like multiple cores of the modern machines.

Algorithm

- Used the minimax algorithm with alpha-beta pruning.
- Machine will make a move by trying out all possible moves for itself and the opponent for next 'x' rounds where x is configurable.
- Machine tries to maximize its revenue and assumes that the opponent will try to minimize the machine's revenue.
- Alpha-beta pruning is used to control the exponential growth of the algorithm by setting limits on maximum and minimum revenues possible thereby reducing the search space.

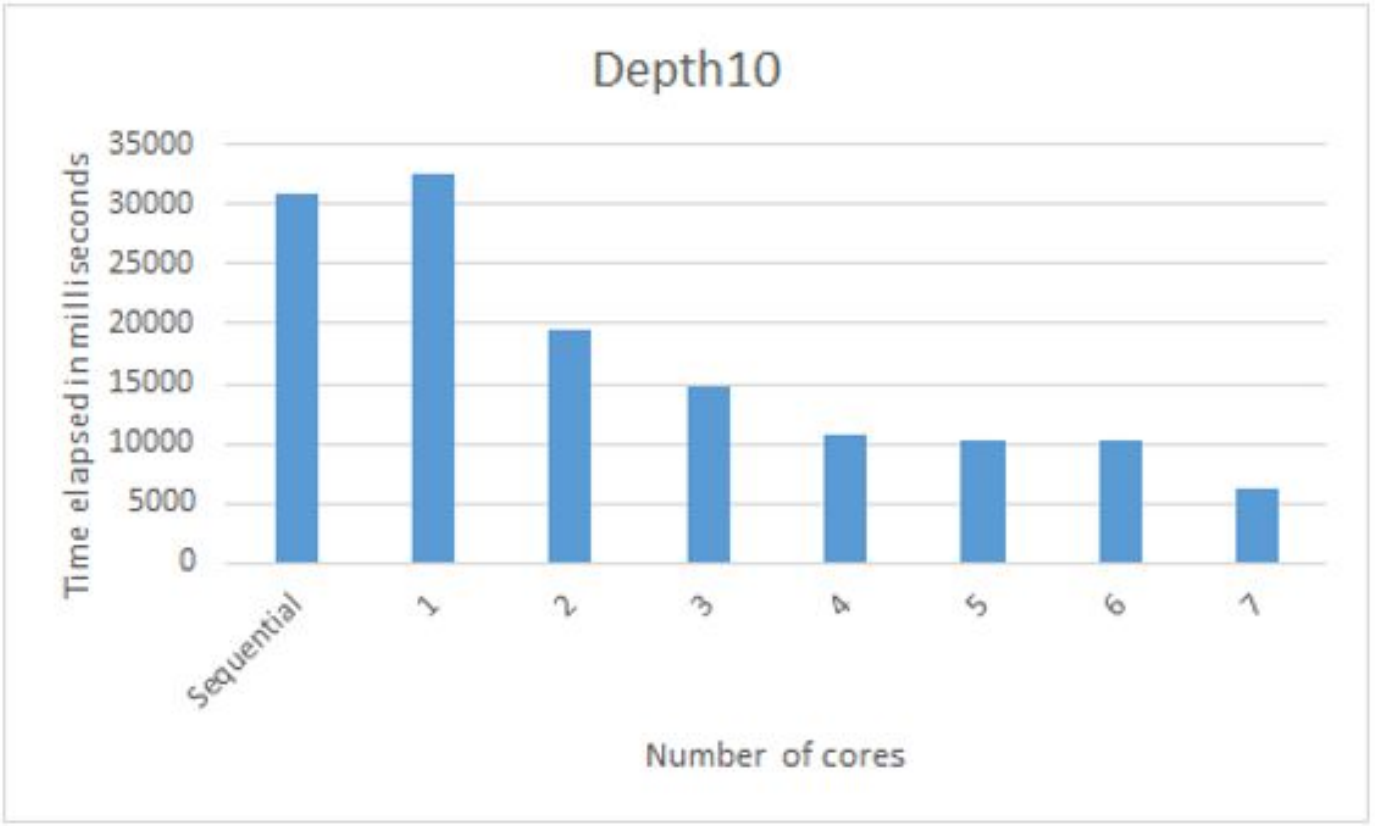
Connect4 with parallelism

- Initially the score for each move is calculated sequentially.
- Problems will arise if depth is considerably greater.
- Performance can be enhanced by calculating the score using parallelism.
- Each move is independent of other moves at a given depth.

Parallelizing using Scala Actors

- In Scala actor system, each actor talk to other actors through ‘messages’.
- We can assign role of a master to an actor and role of a worker to another set of actors.
- Master initializes a worker router which routes the work to different workers through messages.
- Master aggregates the results from all the workers by consolidating the replies from workers through messages.

Results



No.of cores	Standard deviation
Sequential	1061.097807
1	239.5688256
2	713.8995416
3	176.7295486
4	211.7840226
5	64.3801643
6	110.6386762
7	135.143134

Ease of use

- Cores and Threads are configurable.
  - The number of threads that are executing at a particular point of time will atmost be equal to number of cores that are passed as an argument. (Useful when number of threads exceeds the cores).
- Getting a return value from Scala Actors using Messages.
  - When work is given to worker actors, we can get the return value from them using messages. This makes using Scala actors more convenient compared with other languages like Java
- Using futures to get a result from an actor.
  - Using futures, we can specify the caller thread that a result will be given to it in future.

EXTRA

References

[1] https://en.wikipedia.org/wiki/Connect_Four

[2] https://en.wikipedia.org/wiki/Minimax

[3] http://docs.scala-lang.org/overviews/parallel-collections/overview.html

[4]https://en.wikipedia.org/wiki/Scala_(programming_language)

EXTRA

Thank you