Fail-Safe Cloud Tournament Engine with Error Detection and Error Recovery

Kyle Chapman (20703236)
Supervisor: Dr. Cornelia P. Inggs
Cosupervisor: Mr. Andrew J. Collett

February 2020

1 Introduction

The Cloud Tournament Engine (CTE) used in 3$^{\text{rd}}$ year at Stellenbosch University is designed to manage many Othello[1] tournaments, involving many players, concurrently. A user is able to view public tournaments, players, referees, schedulers, etc. and view relevant statistics for ongoing and completed tournaments. The admin, usually the lecturer, is able to add tournaments, schedulers and rankers.

A scheduler is used to schedule matches between two players in a round-robin fashion, where every player plays against every other player. After every player has played against every other player, the overall win-loss ratio can be calculated that can be used to distinguish between good, average and bad players. The win-loss ratio and the Elo[2] rating can be used in conjunction.

A ranker is used to distinguish between the good and bad players by comparing the Elo of the two players, where every player starts on the same amount of Elo. For every win that a player has, they gain a certain amount of Elo and for every loss that a player has, they lost a certain amount of Elo. The amount gained or lost decreases for every match that the player plays, meaning that eventually the Elo rating will be a true reflection of the skill level of the player.

A referee is used to monitor the moves that the two players make in a match and check if any player makes an invalid move or times out. One player sends a move to the referee, which then checks if the move is valid and only sends the move to the other player if the move made is valid. If the move made is invalid, or the player times out, the match will be forfeit and the last player to send a valid move becomes the winner.

The current CTE was used in 2019, but it was found to have some major limitations. The web interface that is used is not user-friendly, there is minimal to no error reporting or error recovery, the code base is umaintainable and it relies on older technology. In many scenerios, a player failure results in a failed tournament when running the CTE.

[1] Often referred to as Reversi.

[2] `https://en.wikipedia.org/wiki/Elo_rating_system`

2 Goals

2.1 Build a user-friendly web interface

The current web interface is very unintuitive as it requires the user to go to many different pages just to upload and add a player to a tournament, amongst other things.

The solution is to follow the three-click rule, where the user can accomplish anything they want to in the interface using a maximum of three mouse clicks. Vue[3] will be used to make the web interface user friendly and visually appealing.

2.2 Have a clean and maintainable code base

The current code base is very messy and requires a lot of searching through the source code to decipher what is being done, meaning that it will become challenging to maintain in the future.

The different aspects of the application will be split into their own files and there will be as much as documentation as possible, so that when a new person looks at the documentation file, they will know exactly what to do.

2.3 Manage each tournament independently

The use of Docker[4] will allow each tournament to be run independently of each other and in their own containers. This will allow multiple tournaments to be run concurrently and not interfere with any other tournaments.

Kubernetes[5] (K8S) will be used to deploy and manage the Docker containers as well as replace the need for Docker swarm. Kubernetes is meant to be scalable and thus a perfect fit for the purpose of the project.

2.4 Error reporting and error handling

The current CTE has minimal error reporting, which makes debugging much harder than it needs to be. There is also no error recovery currently, so when a player crashes, the whole tournament crashes.

There will be as much error reporting as possible and automatic recovery from errors will be implemented as far as possible, so that a player can crash, or timeout, without affecting the entire tournament.

[3] https://vuejs.org/

[4] https://www.docker.com/

[5] https://kubernetes.io/