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

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

9 November 2020

Contents

Chapter 1

Introduction

The Fail-Safe Cloud Tournament Engine (FSCTE) is an extension of the Cloud Tournament Engine (CTE) developed by Reece Murray in 2018, which is an extension of the Tournament Engine (TE) built on the Ingenious Framework[1] (IF). The goal for the CTE is to allow people to play a turn-based game, provided it is supported in the IF, against each other in a match, where many matches make up a tournament. Multiple tournaments can be run to determine how good each player is. Tournaments can separated into two categories: private and public.

Private tournaments are tournaments that only the administrators can add players to, and they are mainly used to test some players against other players. Public tournaments are tournaments that any user can add their own players to and administrators can also add players to. These tournaments are used to put players against each other to rank each player based on how many wins they get against other players.

It was found that there were some major limitations to the CTE, such as minimal error reporting, lack of error recovery and an unmaintainable code base to name a few. As a result, there were situations that arose in which a player failure resulted in a failure of the entire tournament. This prompted the need for a version that was fail-safe and included error reporting and error recovery, as far as possible. The FSCTE aims to address these limitations and improve the overall system stability by being fail-safe, which means that if any error were to occur, the response to the error would cause as little harm to the overall system stability as possible.

[1]`https://bitbucket.org/skroon/ingenious-framework/src/master/`

1.1 Scope

The system used to manage the FSCTE is made up of multiple Docker[2] containers, which run each section of the FSCTE in their own separate environment. If some error occurs in a section of the system, it will not affect the other sections, since they are in separate environments.

The FSCTE uses the following three main sections that are run in their own Docker containers:

- **Web User Interface**: Allows a user to interact directly with the FSCTE.
- **Database**: Stores all the information relevant to the FSCTE.
- **Database watcher (Golang)**: Watches the database for any changes, such as tournaments being started, and sends messages to the front-end accordingly.
- **REST API**: Facilitates file upload to, and download from, the web user interface.

The layout of the entire FSCTE system is further discussed in chapter 3, *design and implementation.*

1.2 Document Outline

This report will discuss the overview of the system in section 2, the implementation and design of the system in section 3, then onto the testing in section 4 and discussion of the future of the project in section 5. Finally, the conclusion will be presented in section 6.

[2]`https://docs.docker.com/engine/docker-overview/`

Chapter 2

System Overview

This chapter will give an overview of the requirements for the FSCTE system and how the requirements have been met.

2.1 Ingenious Framework (IF)

The CTE used in 3^{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 rating [2] 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.

Chapter 3

Design and Implementation

Chapter 4

Testing

Chapter 5

Future Work

Chapter 6

Conclusion

Bibliography

[1] Masters Traditional Games. *Rules and Instructions for Reversi and Othello.* 2019. URL: `https://www.mastersofgames.com/rules/reversi-othello-rules.htm`.

[2] Adam Newell. *What is Elo? An explanation of competitive gaming's hidden rating system.* Jan. 2018. URL: `https://dotesports.com/general/news/elo-ratings-explained-20565`.