A Game Battle Platform based on Web-API for Artificial Intelligence Education

1st Xiaofei Han
Beijing Key Laboratory of Intelligent Telecomm. Software and Multimedia
Beijing University of Posts and Telecomm.
Beijing 100876, China
xfhan@bupt.edu.cn

2nd Junwei Zou
School of Electronic Engineering
Beijing University of Posts and Telecomm.
Beijing 100876, China
buptzjw@bupt.edu.cn

3rd Wei Ren
Beijing Key Laboratory of Intelligent Telecomm. Software and Multimedia
Beijing University of Posts and Telecomm.
Beijing 100876, China
rw478463266@gmail.com

4th Yan Sun
Beijing Key Laboratory of Intelligent Telecomm. Software and Multimedia
Beijing University of Posts and Telecomm.
Beijing 100876, China
sunyan@bupt.edu.cn

***Abstract*—The potential for computer games as a tool for AI research and education continues to blossom. The game battle platform provides a place for students to learn artificial intelligence theory and practice artificial intelligence algorithms. However, the existing platforms perform poorly in terms of possessing high concurrency, debugging bot and supporting Scratch language. In order to resolve these problems, this paper first presents a web-API battle mechanism using Websocket protocol. It increases the number of concurrent matches on servers. Next, we give an offline battle tool to make it easier to debug and test bots for users. In addition, the special environment is introduced to support programming bots in Scratch. Finally, we conduct the comparison experiment between the proposed mechanism and traditional battle methods. The results show that it supports higher concurrency and reduces CPU usage by 40%.**

***Index Terms*—game battle, AI education, Scratch**

I. Introduction

Artificial Intelligence is a rapidly developing field, attracting an increasing number of researchers and learners in the past few years. It is difficult and monotonous for most teenagers to learn about data structures and artificial intelligence algorithms. Currently, the game battle platforms effectively increase the code motivation of teenagers by combining games into the programming and artificial intelligence courses [1]. Teenagers can develop the bot code with intelligent algorithms according to the game rule, and compete against other bots.

Game battle platforms [2] are generally divided into offline and online battle platforms. In the existing work, the offline platform is running the bot on the user's computer. It is difficult for users to compare their bots with others. The online platform is always running the bot on the server, thus the number of concurrent matches is limited. Moreover, the servers always run invalid matches due to the illegal output format of bots, which can cause poor performance of servers. Furthermore, the existing platforms are mainly for high school and college students and do not support the popular Scratch programming [3].

To solve the above problems, this paper proposes a novel game battle platform that combines offline modules and online modules. In addition, it supports multiple programming languages and a variety of interesting turn-based games. Specifically, we first design the web-API [4] battle mechanism using Websocket protocol to improve the number of concurrent matches. When the server is executing a match, bots are running on the user's computer instead of the server. Thus it can support more matches at the same time. Second, we develop an offline battle tool to facilitate to locally debug the bots. It reduces invalid matches incurred by illegal output format in the platform. Third, in order to make young students experience the fun of programming learning, we introduce the special environment to support programming bots in Scratch. Children can develop bots by dragging building blocks to compete with others using other languages. Finally, we conduct a comparison experiment between the proposed mechanism and traditional battle methods. The experimental results prove that the proposed mechanism has high concurrency and low server load.

II. Related Work

There are many forms of game battle platforms. For example, International Computer Games Association (ICGA) is responsible for hosting the International Olympic Computer Games Championship, which is committed to providing an international platform for computer game activities. The game includes 10 items such as Amazon Chess, Gomoku, and Go [5]. Since matches are only made periodically, users can usually only program and test the bots offline. It is difficult for users to learn from each other. Alloyteam develops an online programming game called CodeTank, which is a competitive programming game platform that allows players to learn and improve Javascript programming and AI research [6]. However, the platform can only support JavaScript and the single game. Both Riddles.io [7] and Botzone [8] support multiple languages and a variety of games. The bots on Riddles.io are running on the server, thus a number of concurrent matches will cause the server high CPU usage. In addition, users can not view the match until it is finished to judge. Botzone is developed and maintained by the AI lab of Peking University.

978-1-7281-3129-0/20/$31.00 ©2020 IEEE

Although it supports the API battle mechanism, it is based on the HTTP protocol. You need to check whether there are new messages by polling, which is a waste of resources. And it is complicated to transform the bot into a bot with the web-API mechanism. Furthermore, none of these platforms support the Scratch programming.

In this paper, we first propose a web-API battle mechanism using Websocket. It can solve the problem of low concurrent matches. Second, we develop an offline battle tool which is convenient for users to test and debug bots locally. Third, we introduce the special environment to support for Scratch bots to battle.

III. System Overview

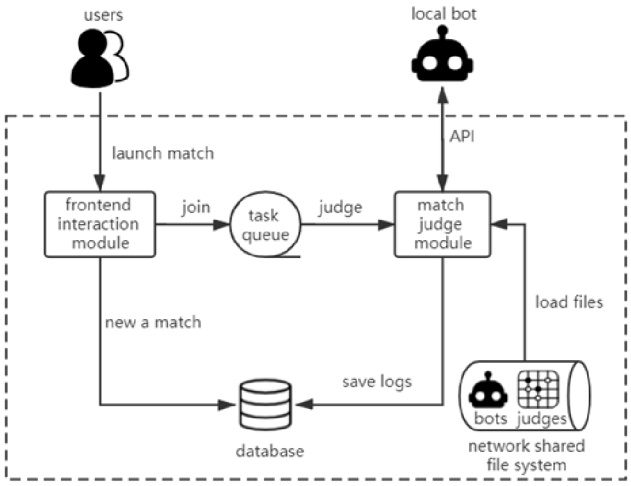

Fig. 1. Major parts of the platform.

As shown in Figure 1, the proposed system is mainly composed of frontend interaction module, match judge module, and data storage module. In the frontend interaction module, users can upload their programs as bots, launch matches, watch live matches, and play back matches. The game judge module act as a core role in the platform. It is responsible for judging the match in the sandbox environment, determining the result of the match, and sending the real-time log of the match to the frontend module. The data storage module includes a network shared file system and a database. The network shared file system is responsible for storing code of game judges and bots, and the database is responsible for storing match records and platform data. The new match enters the task queue, and then the match judge module performs the match. It runs in a sandbox environment, which can isolate the server from the environment of the judge machine. Thus, it can prevent malicious bots from attacking the server. The match judge module runs the game judge and bots required and records the game result. During the match, users can watch the live match on the frontend. After the match is over, the result of the match is saved to the database for playback.

There are two major highlights for the platform: higher concurrent matches and support the Scratch language. First, the game battle platform uses the web-API mechanism to battle and provides bot loader for user experience. The bot loader loads and runs the bot locally, and transmits information in realtime with the server using the Websocket protocol. It greatly reduces the pressure on the server. In addition, users can use the offline battle tool to test and run bots, which can reduce the server to perform some invalid matches and optimizing server performance indirectly. Second, the platform supports a variety of mainstream programming languages including Python, C/C++, Java, etc. In addition, we also support Scratch language for programming bots for teens. Specifically, we introduce a special environment on the basis of Scratch 3.0, which can run .sb3 Scratch file on the terminal. Using this environment makes it possible to perform data interaction with sb3 on the terminal. As shown in Figure 2, the environment is running the Scratch bot of the Gobang game.

```
$ node main.js output.sb3
  action
  {"round":1,"borad":-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
  1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
  ,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
  1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
  ,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
  1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1}
  action {"row":5,"col":6}
```

Fig. 2. Run the Scratch bot in the terminal.

IV. Methods for High Concurrence

A. web-API Mechanism

The architecture of the web-API mechanism is shown in Figure 3, and it is developed based on the traditional battle mechanism. Judge is the code of a game rule that is used to accept bots' outputs and issue instructions to bots. The Websocket server is responsible for sending instructions to the local bot and sending the received information to judge. The match middleware is the core of the web-API engine. It plays the role of the communication bridge between the judge and the Websocket server. The main algorithm of match middleware is shown in Algorithm 1. The whole process can be divided into three steps: running the program (line 2-6), listening for message (line 7-21), and forwarding message (line 23-26).

In step one, the algorithm loads the match configuration file and then runs judge and the Websocket server.

In step two, the algorithm listens for the message of judge and parse the format of the message that is shown in Table 1. The corresponding processing is performed according to different message formats.

In step three, the algorithm forwards the message and records it in the match log.

In order to make it easier for users to use the web-API mechanism to battle, we encapsulate the communication part

TABLE I
MESSAGE FORMAT BETWEEN JUDGE AND LOCAL BOT.

	Instruction	Function
Judge to bot	configuration settings [type] value	Initialize the game configuration
	ask [botId] action [type] [time]	Send command to the bot
bot to Judge	action [actionDetail]	Output decision information

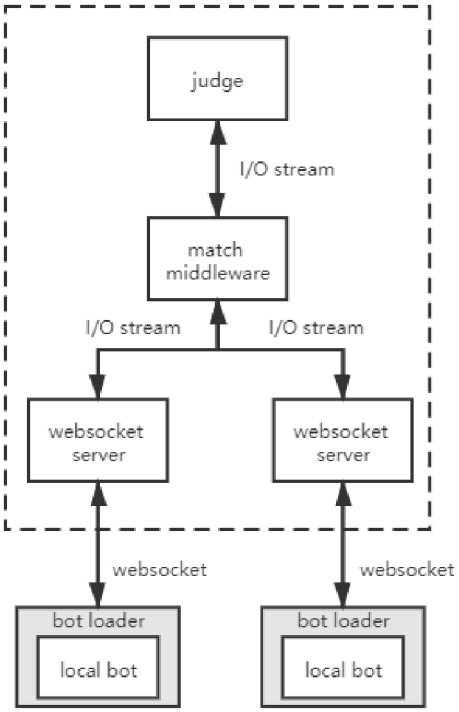

Fig. 3. Architecture of the web-API mechanism.

of Websocket into the bot loader PC program as shown in Figure 4. The core algorithm is shown in Algorithm 2. The bot loader is responsible for forwarding the server information to the local bot and forwarding the output of bots to the server. Users can use the web-API mechanism to battle without adding additional communication codes manually in the bot.

The web-API mechanism can reduce the pressure of the server. In addition, it is possible to make full use of the local CPUs to shorten the execution time of bot. In this paper, the mechanism extends the architecture of the traditional mechanism in the form of a plugin. In the traditional mechanism, the server runs bots instead of Websocket server codes. This extended design allows the platform to support the match between the local-running bot with web-API mechanism and bots with the traditional mechanism.

Algorithm 1 Match Middleware

Input: match config
Output: match result

function JUDGE($config$)
  $result \leftarrow$ ""
  $Judge \leftarrow$SPAWN($config, gameJudgePath$)
  for $i = 0 \rightarrow botPaths.length$ **do**
    $bot[i] \leftarrow$SPAWN($config.botPath[i]$)
  end for
  while $True$ **do**
    $message \leftarrow wait\ message\ from\ Judge$
    if $message$["$type$"] == "$configuration$" **then**
      for $i = 0 \rightarrow bot.length$ **do**
        SENDMESSAGE($bot[i], message$)
      end for
    else if $message$["$type$"] == "ask" **then**
      $id \leftarrow message$["id"]
      SENDMESSAGE($bot[id], message$)
      $message \leftarrow wait\ message\ from\ bot$
      SENDMESSAGE($Judge, message$)
    else
      return $result$
    end if
  end while
end function
function SENDMESSAGE($process, message$)
  $send\ message\ to\ stdin\ of\ process$
  $update\ global\ result$
end function

Algorithm 2 Bot Loader

Input: Url with privateKey, and botCommand
Output: Void

function RUN($url, privateKey, botCommand$)
  $ws \leftarrow$ CREATEWEBSOCKET($url, privateKey$)
  $bot \leftarrow$ SPAWN($botCommand$)
  $listen\ message\ from\ ws$
    SENDMESSAGETOPROCESS($bot, message$)
  $listen\ message\ from\ bot$
    $ws\ send\ message\ to\ server$
end function

B. offline battle tool

We provide an offline battle tool for PC desktop. As shown in Figure 5, the match judge module is the same as the online battle platform, which can restore the environment of the online match. Students use this tool to test and challenge the bot locally. It can reduce invalid matches incurred by illegal output format on the server.

The offline battle tool can not only make it easy to debug, but also allow offline users to experience fun of code fight.

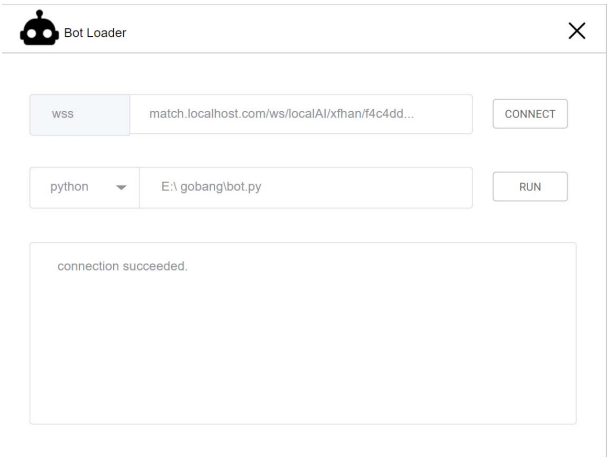

Fig. 4. The bot loader.

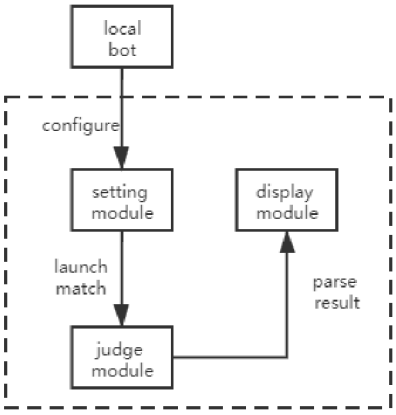

Fig. 5. Structure of the offline battle tool

V. EXPERIMENT

A. Experiment set-up

The experiments are conducted on Windows 10 desktop with 4-core CPU, 16GB of RAM, and running the Chrome browser version 77.0.3865.120. Clients and the server use the same configuration of the computer in the experiment. In addition, we use Celery as the distributed task queue.

In order to evaluate the web-API mechanism, we compare it with the traditional mechanism named cloud bot on the Gobang game. 10 testers used the same bot, and two in groups play against each other. First testers use the cloud bot mechanism to start the match at the same time. After all matches end, the tester runs the bot loader to run the local bot on their computer and then uses the web-API mechanism to start the match at the same time.

In order to make a fair comparison, we need to ensure that the computing environments of all bots are consistent. In the cloud bot mechanism, each CPU core of the server executes a match. Thus, the multiprocessing mode of celery is used to perform the mechanism. In the web-API mechanism bots run on the user's computer which guarantees the computing environment is not affected by the server CPU. Therefore, the web-API mechanism uses the multithreading mode in celery.

B. Sample learning

The game of Gobang is a simple and well-known game played on a 15 * 15 board. Players alternate turns placing a stone of their color on an empty intersection. The winner is the first player to form an unbroken chain of five stones horizontally, vertically, or diagonally [9]. We provide judge code about the rule of Gobang, a game display player, and bot samples in various languages. The bot samples provide the basic data communication code and algorithm logic for the bot developers. In addition, we develop some building blocks for the Scratch samples of Gobang. It is convenient for the user to focus on the development of the core algorithm. The Scratch 3.0 bot sample for the Gobang game is shown as Figure 6. Player 1 develops a bot based on the Scratch sample provided

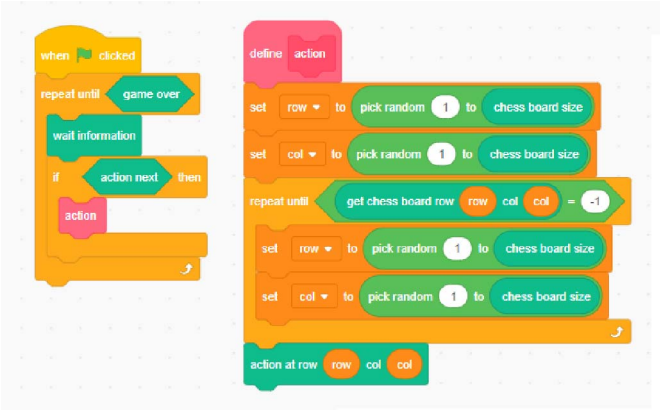

Fig. 6. Scratch bot sample.

by the platform, and Player 2 develops a bot based on the Python sample. After the development, the two players use the bot loader to enter the private key and the command to run their bots, and join the same room to battle. Then they can watch the real-time match on the HTML5 player.

C. Performance analysis

TABLE II
RESULTS BETWEEN THE TWO MECHANISMS.

Characteristic	cloud bot	web-API
total time to complete all matches	145s	97s
average CPU usage	66.3%	21.0%
average round time	1015ms	1067ms
average match time	71.1s	74.0s

The experimental result is shown in Table 2, which proves the web-API mechanism has higher concurrency than the cloud bot method. When multiple matches are launched at the same time, the server does not start executing a new match task until there is an idle CPU. However, in the web-API

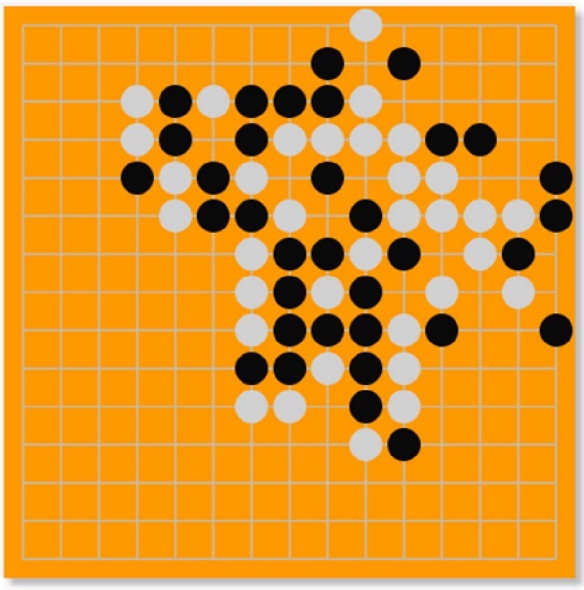

Fig. 7. A situation of Gobang game.

mechanism in multithreading, match tasks can be executed simultaneously. Due to the communication overhead between the local bot and the server, the average round time of bots and the average match time in web-API mechanism are more than the cloud bot method.

D. Platform comparison

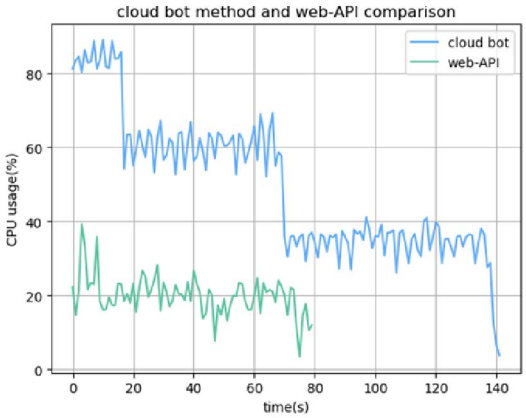

Fig. 8. Comparison of result between the two mechanisms.

TABLE III
COMPARISON WITH OTHER PLATFORMS.

Characteristic	Riddles.io	Botzone	Our platform
Local match tools	Yes	No	Yes
API Protocol	/	Http	Websocket
Local bot loader	/	No	Yes
Support Scratch bot	No	No	Yes

The comparison between our platform and others is shown in Table 3, which proves there are great advantages in our platform. It supports web-API mechanism using Websocket protocol and Scratch language. In addition, we develop a bot loader to make users easily battle with the proposed mechanism. Overall, this platform provides high concurrency and is simple to use.

VI. Conclusions

In summary, we propose a game battle platform for AI education that supports offline battle and online battle of high concurrency. In addition, it supports a variety of programming languages including Scratch. Specifically, we first design the framework of the web-API mechanism which can improve concurrency in the execution of matches. Second, we develop the offline battle tool to assist users in debugging and testing bots locally. Third, we introduce a special environment to support Scratch to develop bots for teens. Finally, we conduct a comparative experiment between the web-API mechanism and the cloud bot mechanism. The results show that the web-API mechanism is higher concurrent and takes up less resources on the server. The limitation of the web-API mechanism is only helpful for the matches played by both players using the local bot loader.

In the future, we will propose a better mechanism to reduce server load no matter how the players participate. We will also support ranking mode, in which students can see their ranking in the leaderboard.

Acknowledgement

This work is partly supported by the National Natural Science Foundation of China under Grant 61877005, 61672109 and 61772085.

References

[1] M. Papastergiou, “Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation,” Computers & Education, vol. 52, no. 1, pp. 0–12.
[2] F. Lu, K. Yamamoto, L. H. Nomura, S. Mizuno, Y. Lee, and R. Thawonmas, “Fighting game artificial intelligence competition platform,” in 2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE). IEEE, 2013, pp. 320–323.
[3] M. Resnick, J. Maloney, A. Monroy-Hernandez, N. Rusk, and Y. B. Kafai, ´ “Scratch: Programming for all,” Communications of the Acm, vol. 52, no. 11, pp. 60–67, 2009.
[4] H. Zhou, H. Zhang, Y. Zhou, X. Wang, and W. Li, “Botzone: an online multi-agent competitive platform for ai education,” in Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. ACM, 2018, pp. 33–38.
[5] ICGA, “International computer games association,” https://icga.org/, accessed January 4, 2020.
[6] AlloyTeam, “Codetank,” http://codetank.alloyteam.com/, accessed December 10, 2019.
[7] J. van Eeden, “Riddles.io,” https://riddles.io/, accessed June 18, 2019.
[8] H. Zhang, G. Gao, W. Li, C. Zhong, W. Yu, and C. Wang, “Botzone: A game playing system for artificial intelligence education,” in Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS). The Steering Committee of The World Congress in Computer Science, Computer . . . , 2012, p. 1.
[9] Wikipedia, “Gomoku-wikipedia,” https://en.wikipedia.org/wiki/Gomoku, accessed December 20, 2019.