Teaching Introductory Parallel Programming using Two-Player Online Games

Sathiamoorthy Manoharan and Xinfeng Ye
School of Computer Science
University of Auckland
New Zealand

***Abstract*—Parallel programming is a powerful tool for computing tasks faster and more efficiently. Teaching an introduction to parallel programming requires explaining the fundamentals of parallelism, the various types of parallel programming constructs, and the tools and techniques used to build and debug parallel programs. Teaching introductory parallel programming can be a challenging task. This paper is an experience report, detailing how a two-player online game is built as a motivating focal point to engage students and explain various parallel programming concepts and constructs.**

***Index Terms*—teaching programming, parallel programming, asynchronous programming, two-player games**

I. Introduction

The benefits of parallel programming are readily realized in the modern computing landscape where processors have many cores. It is a powerful technique that can be used to drastically improve the performance of applications and make them more efficient. As such, it is essential for students to have a good understanding of parallel programming techniques in order to best utilize the technology.

A parallel program course would typically cover many fundamental concepts such as threads, processors, cores, distributed systems, and synchronization. It would compare single-threaded and multi-threaded programming, and explore the various types of parallel programming such as data parallelism, task parallelism, and pipeline parallelism.

For an introductory course on parallel programming, some of these concepts can be difficult to learn without hands-on practical work. We took a project-based learning [1] approach whereby the students built a two-player online game from the ground-up to to gain a deeper understanding of the concepts and apply their knowledge in a practical setting.

This paper is an experience report that discusses the benefit of developing a two-player online game using a project-based learning approach in achieving the learning outcomes of our introductory course on parallel programming in the School of Computer Science.

The rest of the paper is organised as follows: Section II reviews some of the recent work related to ours. Section III outlines the project as specified, while section IV discusses how the project was executed. Section V adds further considerations and reflections. The final section concludes with a summary.

II. Related Work

This section discusses some of the recent related work in teaching parallel programming courses. Earlier works are reviewed systematically by Carneiro Neto and colleagues [2].

Younis and colleagues [3] use a series of multi-themed group projects to introduce parallel programming to their computer organization class. They report results showing significant improvement to the students' parallel programming skills, which they attribute to their use of project-based learning.

Chen reports on the design of a parallel programming course for junior students, drawing from prior expertise in teaching senior students [4]. Nearly half of the course is devoted to programming exercises, offering students plenty of hands-on experience.

Danelutto and Torquati leverage structured parallel programming principles and several parallel programming frameworks to improve the teaching and learning process [5]. They argue that over a period of eight years, their teaching approach has shown a steady trend of increase in student performance as implied by the student grades.

Martins and colleagues use programming-contest-style challenges in parallel programming to encourage students to learn parallel programming [6]. They argue, using quantitative and qualitative scores, that such challenges engage students and motivate them to master the concepts required to solve the challenges.

Marzulo and colleagues, just like Martins and colleagues, use challenges to teach parallel programming [7]. In this case, the challenges are parallel programming marathons, which, according to the authors, kindle student interest and increase motivation.

Kurniawati presents a short paper on their experience with teaching parallel programming, and notes the difficulty students faced in designing and debugging parallel programs [8].

Conte and colleagues stress the importance of the teaching parallel programming early in the degree program so that students have the background and skills to use parallel programming concepts throughout their degree [9].

III. The Project

The project required students to design and implement a multithreaded game server suitable for two-player games (such as Backgammon, Checkers, Chess, Go, etc.). The server helps to pair up players and to coordinate the exchange of game

moves. It does not know anything about the underlying game itself. In other words, the game server does not attempt to understand the game moves or enforce any rules. The game rules are left to the two players to enforce, just like in a real board game. This allows the game server to be generic, and enables the students to focus their design and development solely on the multithreading aspects.

The server was to be programmed using the C# programming language, and needed to be constructed using a synchronous server socket from the ground up. Higher-level APIs (such as HTTPListener) were not allowed, and the students had to implement the required multi-threading and synchronization themselves.

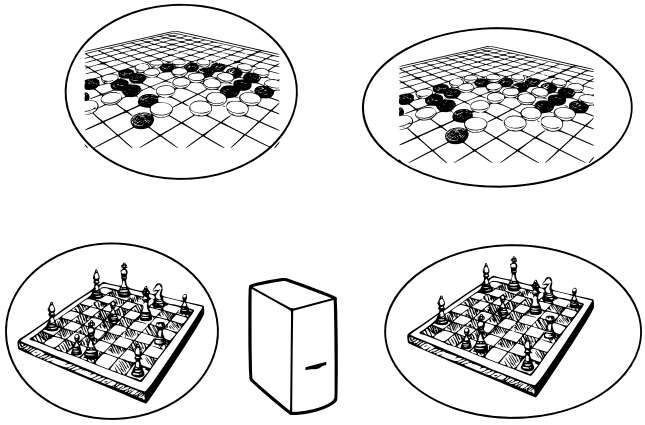

Fig. 1: An illustration of the multithreaded game server for two-player games. The game server does not attempt to understand the game moves or enforce any rules. The game rules are left to the two players to enforce, just like in a real board game. This allows the game server to be generic, and enables the students to focus their design and development solely on the multithreading aspects.

The server was to use HTTP REST as the basis for communication with the game clients. To this end, the following (GET) endpoints were suggested:

/register
: This endpoint generates a random username for a player, registers this name, and returns to the user the registered name. The player is required to pass this username in all subsequent transactions for identification.

/pairme?player={username}
: This endpoint attempts to pair the given player with another player. It returns a game record (or a suitable subset of it). The game record is a tuple containing a game ID, game state, username of the first player, user name of the second player, last move of the first player, and the last move of the second player. When there is no other player waiting, the game record will contain a newly allocated game ID (which could be a GUID), a game state indicating ”wait”, and the username of the requesting player as first player. The rest of the elements in the tuple are not defined. When there is a waiting player, the game record will be the game record first created for the waiting player, updated to add the second player and the state indicating ”progress”. The state ”progress” tells both players that the game can now begin. The endpoint can be invoked by both players as many times as they want before the commencement of the game. This helps the first player to ”poll” the state to see if a second player has been paired up.

/mymove?player={username}&id={gameId}&move={move}
: This endpoint, when used during the game’s ”progress”, will supply the user’s move to the server. The ”last move” of the player in the corresponding game record will be updated with this supplied move.

/theirmove?player={username}&id={gameId}
: This endpoint, when used during the game’s ”progress”, will collect the other player’s move from the server. The game server will supply the ”last move” of the other player from the game record corresponding to the given game ID.

/quit?player={username}&id={gameId}
: This endpoint notes to the server the intention of the player to quit the game. The server will remove the game record corresponding to the given game ID. Attempt by the players to access the game record (for example, to get a move) will fail after the record is removed.

Students were told to implement other endpoints for diagnostic or informational purposes (e.g., a /debug endpoint or a /version endpoint) as required.

They also needed to take appropriate actions when errors are encountered (e.g., invalid endpoints, malformed endpoints, invalid parameters to endpoints, etc.).

Typically, a single thread will handle a single player’s endpoint interactions. The server, for the sake of efficiency, should keep the connection alive to handle the player’s requests in the same thread. However, it is possible that the client may, from time to time, close the connection. For example, a browser will close the connection if it deems the connection inactive for a period of time. The server should be able to gracefully handle this scenario and continue to serve the client (potentially using a new thread).

Students were asked to think about what information is shared among the threads and avoid potential race conditions efficiently using appropriate concurrency controls.

In order not to complicate the server implementation more, students were asked not to persist any of the information. Meaning that, there was no database backend involved – the information is kept only in nonpersistent memory and lost if the server is restarted.

One thing that the server specification did not include is the ability to support multiple game types. This was intentional, as we wanted to see if any student picks this up (and none did).

A. Testing

The students were asked to come up with an appropriate testing strategy for verifying the correctness of the server. They

were required to build a browser-based client for a chosen two-player game to help with their testing and verification plan, and to demonstrate game play using their server.

B. Learning Outcomes

The parallel-programming-related learning outcomes of this project are the following

1) being able to create and use multiple threads
2) understanding critical sections
3) understanding and using mutexes and condition variables
4) avoiding deadlocks
5) ordering events

The marking rubric took into account these learning outcomes. A high mark in the project therefore is likely to indicate that the student met most of the learning outcomes.

IV. Project Execution

Most projects require one to have a broad knowledge of many concepts to execute and complete them successfully. In addition to parallel programming skills, the students required good classic programming skills, an understanding of network programming, in particular socket programming, basic knowledge of HTTP, and systematic testing skills. While most of these topics had been covered in prerequisite subjects the students had studied, we observed gaps in their knowledge as well as lack of practice becoming a barrier to the successful completion of the project.

The key skills a large proportion of students lacked are incremental development and effective testing strategies, both of which are prerequisite material and are well outside of the scope of this subject.

This required us to have additional help sessions to guide students on iterative development. In particular, for this project, we outlined an iterative development plan as follows.

1) Compile and run the synchronous server socket example from the documentation. You will see that the server closes the connection soon after sending the response, and telnet client will terminate since the server closed the connection.
2) Now change the server so that it does not close the connection soon after sending the response. When you have successfully done this, you can use the same telnet client to send requests to the server and receive responses. This connection is now alive, and this is what HTTP's keep-alive does (loosely speaking, since HTTP has a timeout).
3) Now that you have a server that keeps the connection alive, test with another concurrent telnet client (while your first one is still active and connected to the server). You would see that the second telnet client would connect to the server but it won't receive any response to requests it sends. The first client is keeping the server busy here.
4) Try closing the first client and see if the server now responds to the second client. You will see that it won't. This is because the server does not see that the first client has closed the connection. You need to find out how to detect this and get the server to be ready to serve the second client when the first client quits.
5) At this point you have a server that keeps the connection alive and is able to tidy itself up when clients disconnect from the server.
6) Now you are ready to get multithreading into the play to make the server service multiple clients concurrently.

Yet, a number of students did not test their implementation fully. In addition, some of the students assumed that an HTTP request would come in a single packet, an assumption that would not hold if a slow client such as telnet were to be used. This resulted in the average project mark being 48%, less than the pass mark of 50%. We therefore gave everyone a second chance to test and resubmit their project, after providing student-specific feedback in one-to-one sessions. Fig. 2 shows each student's mark for their initial submission as well as their second attempt.

Note that, the class size is small (just 28 students, as can be inferred from Fig. 2), and therefore having one-to-onse feedback sessions with students was possible.

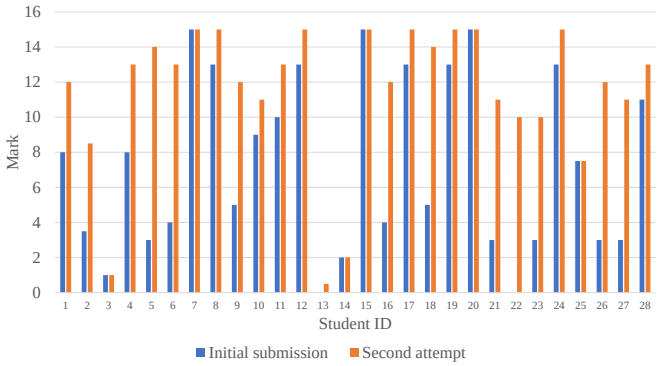

Fig. 2: Initial marks and marks for the second attempt for each student. The project was marked out of 15 and contributed to 15% of the final mark.

On average, students gained 4.2/15, or 28%, marks in their second attempt. If we exclude the three students who already scored full marks in their initial submission, then the gain becomes 4.7/15, or 31%. The average project mark after the second attempt rose to 76%. These observations lead us to believe that the feedback from the initial submission contributed to better marks in their second attempt, and consequently a better overall learning.

Seven students did not use proper synchronization or locking in their implementation. One of the students had a demonstrable deadlock in their game play.

Most students spent a lot of effort to complete the project to a high standard, and informal feedback from the students indicated that they highly appreciated the project-based learning approach and being able to develop a complete game from the ground up. Being generic, the game server allowed students to implement a game client of their choice. Students therefore implemented a variety of good, and working game clients. Fig. 3–Fig. 6 show screenshots of some of these clients.

Fig. 3: A chess client implementation.

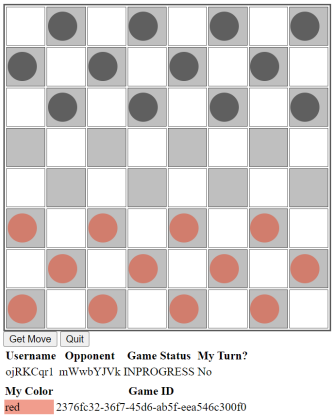

Fig. 4: A checkers client implementation.

V. Discussion

Teaching introductory parallel programming can be a difficult task, especially for those who are new to the subject. There is a lot of material to cover, and it can be difficult to decide what topics to focus on and how to best approach them.

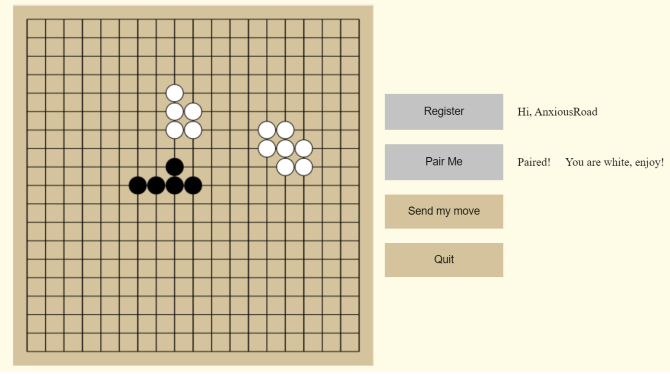

Fig. 5: A gomoku client implementation.

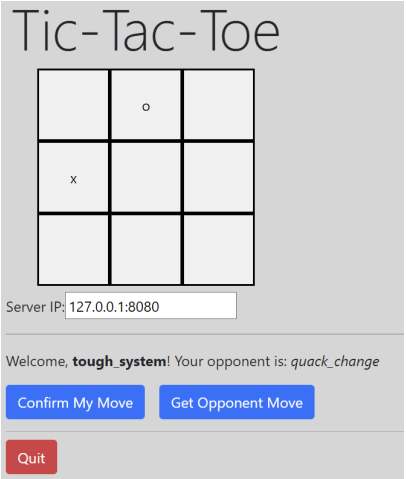

Fig. 6: A tic-tac-toe client implementation.

A practical project such as the one used in this paper provides students with opportunities to explore and extend the topics discussed in class. It helps them gain a better understanding of how parallel programming works in practice in a wide context. Formative feedback and fostering collaborative discussions during the project also help.

Most of the difficulties a number of students faced did not arise from the main focus of the course. They had difficulties with incremental development, testing, debugging, and also lacked network programming skills. While project-based learning is fun and effective, it requires students to have a broad understanding of many related subjects in order to successfully complete the project.

VI. Conclusions

Teaching introductory parallel programming can be a challenging task. However, by providing hands-on experience, and giving students opportunities to explore, it is possible to create an effective and engaging course. Giving students a chance to re-work their practical work after feedback is also a strategy to improve the overall gain in learning outcomes.

References

[1] G. E. Veselov, A. P. Pljonkin, and A. Y. Fedotova, "Project-based learning as an effective method in education," in *Proceedings of the 2019 International Conference on Modern Educational Technology*, ser. ICMET 2019. New York, NY, USA: Association for Computing Machinery, 2019, pp. 54–57.

[2] J. A. Carneiro Neto, A. J. Alves Neto, and E. D. Moreno, "A systematic review on teaching parallel programming," in *Proceedings of the 11th Euro American Conference on Telematics and Information Systems*, ser. EATIS '22. New York, NY, USA: Association for Computing Machinery, 2022.

[3] A. A. Younis, R. Sunderraman, M. Metzler, and A. G. Bourgeois, "Case study: Using project based learning to develop parallel programing and soft skills," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2019, pp. 304–311.

[4] X. Chen, "Designing a parallel programming course for lower-division students," in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2020, pp. 1009–1011.

[5] M. Danelutto and M. Torquati, "Increasing efficiency in parallel programming teaching," in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2018, pp. 306–310.

[6] G. Martins, P. S. Lopes de Souza, D. Jose Conte, and S. M. Bruschi, "Learning parallel programming through programming challenges," in *2020 IEEE Frontiers in Education Conference (FIE)*, 2020, pp. 1–9.

[7] L. Marzulo, C. Bianchini, L. Santiago, V. Ferreira, B. Goldstein, and F. França, "Teaching high performance computing through parallel programming marathons," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2019, pp. 296–303.

[8] R. Kurniawati, "Teaching parallel programming with Java and Pyjama," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*, ser. SIGCSE 2022. New York, NY, USA: Association for Computing Machinery, 2022, p. 1109.

[9] D. J. Conte, P. S. L. de Souza, G. Martins, and S. M. Bruschi, "Teaching parallel programming for beginners in computer science," in *2020 IEEE Frontiers in Education Conference (FIE)*, 2020, pp. 1–9.