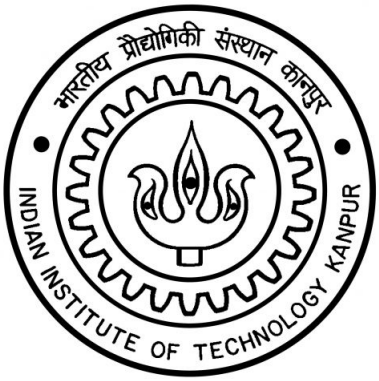
Gambit VSCode Extension

Suvrat Pal(211089)
under the guidance of Dr. Subhajit Roy and Dr. A.K Singh

April 24, 2024



Indian Institute of Technology, Kanpur

Acknowledgement

I would like to express my heartfelt gratitude and appreciation to all those who have contributed to the successful completion of our project, **GAMBIT VSCode Extension**. I extend my sincere acknowledgments to Dr. Subhajit Roy, my esteemed guide from the Department of Computer Science and Engineering and co-supervisior Dr. Amarendra Kumar Singh from Department of Materials Science and Engineering. There guidance, expertise, and unwavering support have been instrumental in shaping this project. I would also like to extend my thanks to the Ph.D. scholar, Pankaj Kumar Kalita, for his valuable assistance throughout the project. Pankaj's expertise, constructive feedback, and dedication have greatly contributed to the overall quality of my work.

Contents

Abstract

Debugging concurrent programs poses unique challenges due to issues such as interleaving and relaxed memory consistency models. This project introduces the GAMBIT VSCode Extension which provide a user-friendly interface with features like memory model selection and query input to facilitate effective debugging of threaded programs. With a focus on identifying and resolving issues stemming from interleaving and memory consistency discrepancies, GAMBIT extension provides a comprehensive solution for debugging concurrent programs, aiming to enhance the efficiency and accuracy of software development in multi-threaded environments.

1 Introduction

Debugging is the process of identifying and resolving errors or bugs in computer software. Debugging involves finding the root cause of unexpected behavior, glitches, or malfunctions in a system and then correcting those issues to restore normal operation.

1.1 Concurrency and Interleaving

Concurrency means many computation happening at the same time. Scheduler schedules which instructions to execute. Instructions from different thread overlap with each other and create *interleaving*. Bugs due to interleaving are difficult to detect and fix.

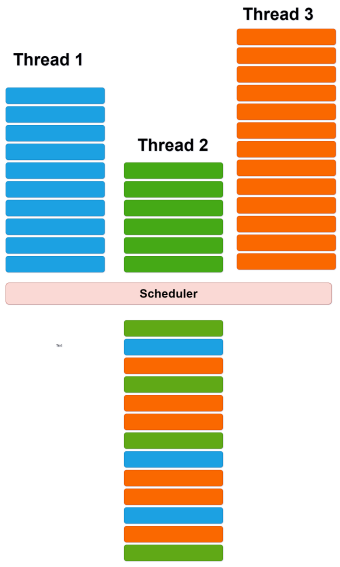


Figure 1: Schedular for multiple threads

Consider two threads being executed. For the first interleaving thr2 is executed before thr1, so the assertion a==b is not violated.

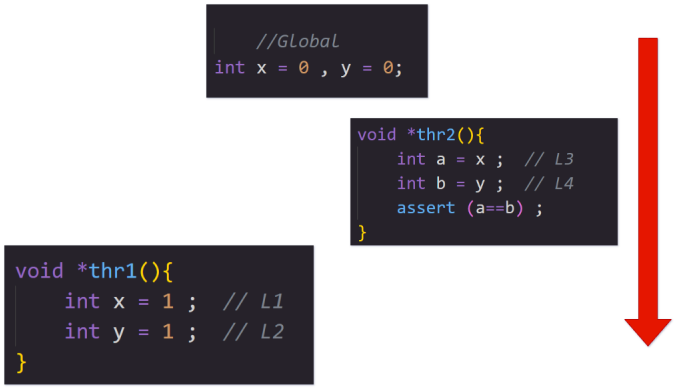


Figure 2: first interleaving

However, for the second interleaving, a=0 and b=1, so the assertion a==b is will be violated.

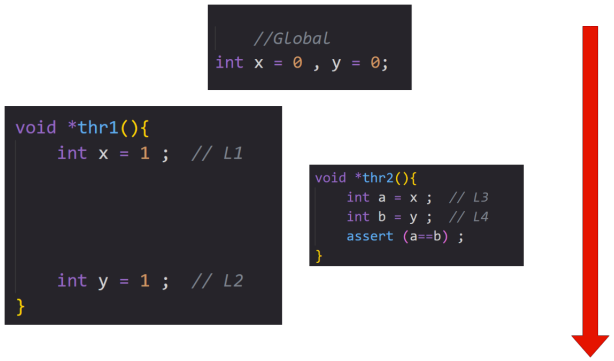


Figure 3: Second Interleaving

1.2 Memory consistency models

Memory Consistency is the problem of defining how parallel threads can see their shared memory state.

- *Sequential consistency(SC)*
  - Execute in program order.
  - Terribly slow.
- *Total store ordering (TSO)*
  - Little relaxed than SC.
  - Store-Load reordering possible of distinct memory locations.
  - Uses store buffer.
- *Partial Store Ordering (PSO)*
  - More relaxed than SC and TSO.
  - Store-Load and Store-Store reordering possible of distinct memory locations.

Hence, there can be many issues in concurrent programmes i.e., bugs due to interleaving, bugs due to relaxed memory model.

1.3 GDB Assisted Memory Behaviour and Interference Tester(GAMBIT)

GAMBIT offers an interactive debugging environment tailored for concurrent programs, aiding in the detection of bugs resulting from memory consistency model discrepancies and interleaving issues[3].

GAMBIT has 3 components:

- GDB
- Symbolic Execution
- SMT Solver

2 GAMBIT VSCode Extension

The extension[1] offers a comprehensive set of tools to debug threaded programs effectively. It includes features such as memory module selection, query selection, variable input, log file creation, modified file generation, and Docker integration for output retrieval.

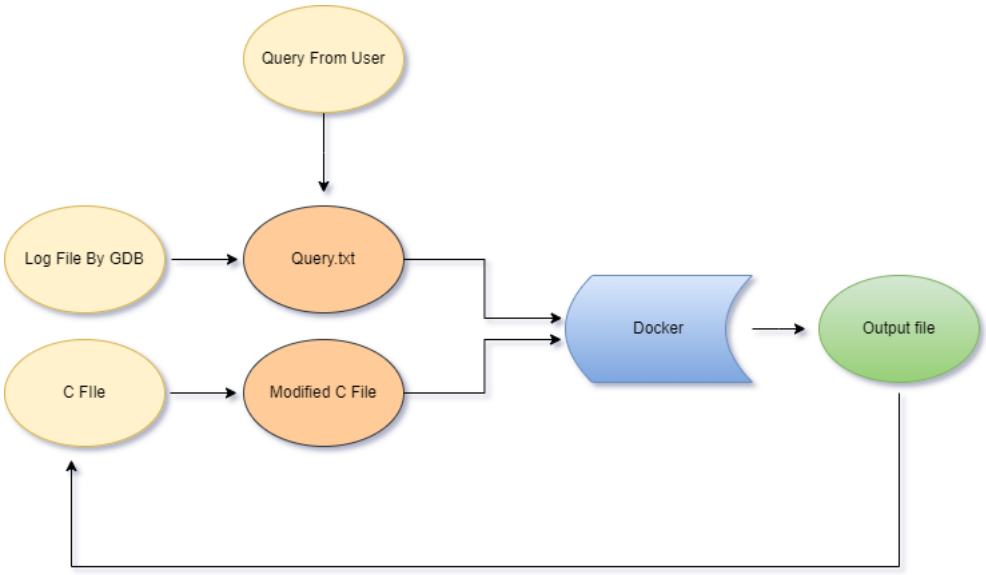


Figure 4: Work Flow of Extension

2.1 Functionalities and Features

- *Activate Button*: Initiates the trace function, enabling users to start the debug session.
- *Dropdown1(Memory Model Selection)*: Allows users to choose the memory module to focus on during debugging.
- *Dropdown2(Query Selection)*: Enables users to select specific queries related to the chosen memory module.
- *Input Text Area*: Allows users to input variable names and their corresponding values for debugging purposes, if required in the query.

- *Submit Button*: Submit Button: Triggers the creation of a folder containing the log file and modified C file. Additionally, it runs the Gambit Docker, which generates a JSON file as output.
- *Output Text Area*: Displays the output generated by the Gambit Docker, facilitating analysis and debugging.

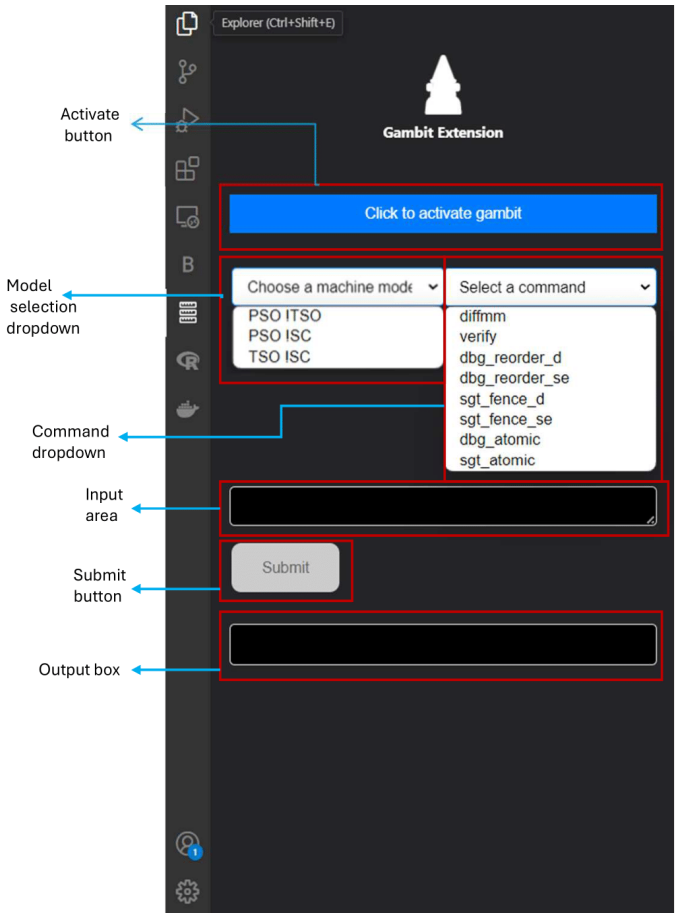


Figure 5: extension screenshot

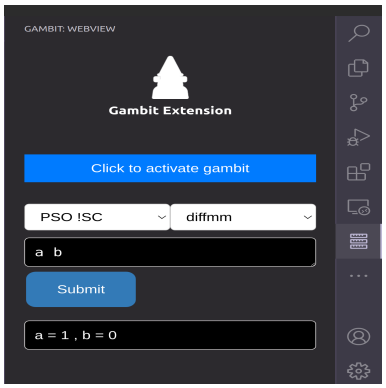
3 Demonstration

Consider the code snippet given along side, containing two threads namely thr1 and thr2 and breakpoints at both the thread functions. We will use the extension to debug the code[2].

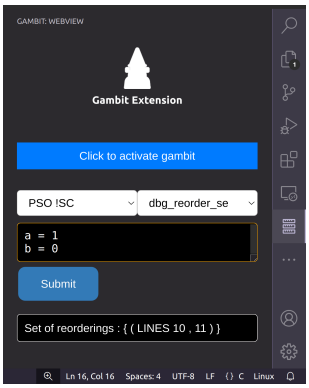
```
 6  int x = 0 , y = 0;
 7
 8  // Thread 1
 9  void *thr1(){
10      x = 1;
11      y = 1;
12  }
13
14  // Thread 2
15  void *thr2(){
16      int a = y ;
17      int b = x ;
18      assert (a==b) ;
19  }
```

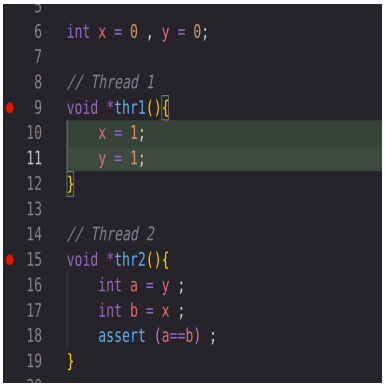
Following examples are demonstrated:

- ***Is there any value for a and b possible in PSO but not in SC?***
  First we run the C file after activating the extension. Due to the breakpoint at thr1 program stops at line 10. Now we execute the two instrunctions of this thread. Now we switch to thread2 an execute the next two instructions. Now at this program point we want to know, can a and b get any weird values? So we select the memory model and command from the dropdowns and enter the variables' name through input panel and push the submit button. The output will be shown in the output box displaying the values a and b can have in PSO model but not in SC.

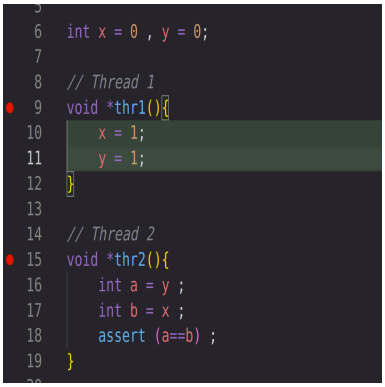


- ***Reorderings in PSO but not in SC such that a=1 and b=0***
  Similary as before, we enter the query and press the submit the button. Output box will display the set of redordering of lines which occur during execution and the particular lines will be highlighted.





- ***Suggest repairs to fix reorderings which produce a=1 and b=0***
  The extension suggests to insert store-store fence between Lines 2 and 3. Due to the insertion of store-store fence at these lines store operations across these fence cannot be reordered. So the weird behaviour due to reorderings will not appear anymore.



4 Conclusion

The Thread Program Debugging Extension offers a comprehensive solution for debugging threaded programs, providing developers with the necessary tools to identify and resolve issues efficiently. Its intuitive interface and robust functionality make it a valuable asset for debugging tasks.

References

[1] https://code.visualstudio.com/api/get-started/your-first-extension

[2] https://www.youtube.com/watch?v=tzkrxFpK-FQ

[3] Verma, A., Kalita, P.K., Pandey, A., Roy, S.: Interactive debugging of concurrent programs under relaxed memory models. In: Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization. p. 68–80. CGO 2020, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3368826.3377910, https://doi.org/10.1145/3368826.3377910