CS406 Project Progress Report

Baturalp Kabadayı, Atakan Saraçyakupoğlu

April 2023

1 Introduction

Parallel programming is a crucial field of computer science, allowing high-performance computing through the simultaneous use of multiple processing units. It is a field that requires exquisite software engineering skills and attention to detail in coding.

Recent advancement in NLP tools such as ChatGPT has created an opportunity in this field for increasing the efficiency of software engineers and computer scientists by allowing them to check errors, problems and possible improvements in their parallel programming software by classifying and fixing code.

Our project aims to capitalize on the potential of NLP tools like ChatGPT by automating quality control processes in parallel programming. By using these advanced tools, we can significantly enhance the efficiency and effectiveness of parallel programming, ultimately leading to better software products and more streamlined development workflows.

2 Progress

First of all, we wanted to use the ChatGPT API in order to automate sending prompts to ChatGPT and processing responses. However, using the API required a paid subscription; therefore, we have postponed the automatization process, and decided to have some initial tests manually.

2.1 Data Preparation

To start our tests, we needed a list of categories and subcategories of HPC performance bugs. For this purpose, the list at the end of this document is prepared, from the mentioned paper in the project document. As it is seen in the list, there are 10 categories, and for some of them, there are subcategories.

Another aspect of testing is a labeled dataset of performance bugged codes. Dataset from the given paper that contains various bug fix GitHub commits and their relevant category and subcategory is used to accommodate this need.

2.2 Methodology

At first, we gave GPT GitHub commit links, and wanted it to classify bug types. However, this method did not work because of the fact that ChatGPT is not able to access something on the internet dynamically. Therefore, this method is switched to giving the source code directly in the prompt. One deficiency about this method is that there is a maximum character limit for the prompt so that one cannot want ChatGPT to analyze and identify the bug in a long source code file.

Since the dataset had the GitHub commit links, not the files themselves, an automatization process has been made so as to get the file content of a given commit. For this purpose, PyGithub is used to interact with GitHub API and get the related content. Being found the related content provided in the commit through the API, the parent of that commit is accessed. The reason for accessing the parent is that HPC bug(s) are fixed in the child commit; therefore, the parent commit has the performance bugs. Finally, the source code is returned.

```python
def get_source_code(link):
    # Last 40 characters of the link is commit sha.
    sha = link[-40:]

    # Repository organization and name:
    repo = link[19:-48]

    # g is the github object created by API key.
    repo = g.get_repo(repo)
    commit = repo.get_commit(sha=sha)
    old_sha = commit.parents[0].sha
    path = commit.files[0].filename
    old_contents = repo.get_contents(path=path, ref=old_sha)
    old_code = old_contents.decoded_content.decode("utf-8")
    return old_code
```

Figure 1: Getting Source Code Using PyGithub

The final step to start tests is to have a question prompt that ChatGPT can easily understand and process. Firstly, we gave the HPC performance bug list to GPT once and then asked it to classify the performance bug in a source code in a different prompt. This method did not work properly because of the fact that ChatGPT was forgetting the list between different prompts. Therefore, it has been decided to give the HPC performance bug list in every prompt. Another automatization tool is created to generate a prompt from a GitHub commit link. Following is the prompt template that is used.

```
Following code has some performance related bugs.
Please identify the performance bugs category and subcategory
from the given list (pick only one):
LIST:
<HPC Performance Bugs List>

FILE:
<Source Code>
```

```python
def generate_prompt(link):
    # Automatic prompt generator from a github commit link.
    prompt = QUESTION
    prompt += LIST
    prompt += "\n"
    prompt += "FILE:\n"
    code = get_source_code(link)
    prompt += code
    prompt += "\n\n"

    # Write the prompt into a file.
    fname = "./data/prompt.txt"
    with open(fname, "w") as file:
        file.write(prompt)
```

Figure 2: Generating Prompt

2.3 Initial Tests

10 commits are sent to ChatGPT for classification, where it had 2 tries to guess. After every response from GPT, an appropriate feedback is given from the following list:

```
* You are correct!
* Incorrect category and subcategory, please try again
* Category is correct but subcategory is not correct, please try again
* Incorrect, it must have been: <correct bug type>
```

Here are the initial test results (GitHub links can be achieved by the RQ1.xlsx, and index) Entries in Actual is the actual category. Category and Subcategory column indicates whether the classification is correct. Moreover, False1 and False2 columns are guesses of ChatGPT for respectively try 1 and try 2. Finally, Error column states that GPT is unable to classify for the demonstrated reason.

		First Try		Second Try		Wrong Guess		
Index	Actual	Category	Sub-Category	Category	Sub-Category	False1	False2	Error
46	1.2	True	False	True	True	1.1		
79	3.2	True	False	True	True	3.1		
64	2.2	False	False	True	True	1.1		
15	1.1	False	False	False	False	2.2	6.4	
172	7.2							Needs more Context
144	6.3							Claims that there is no bug
143	6.3							Claims that there is no bug
39	1.2	False	False	False	False	6.2	6.4	
56	1.2	False	False	True	False	1.2	6.2	
78	3.2	False	False	False	False	1.1	4.3	

Figure 3: Test Results

3 Future Plans

We plan on modifying our project such that it is able to send lengthy source code to ChatGPT API, and gets a response with an HPC Performance Bug Type automatically using OpenAI API. The response will be processed and stored in a results file.

In the second stage of our future plans, we plan on receiving fixed code from ChatGPT API. The code which is fixed by GPT is going to be ran on Nebula machine and based on the performance improvement, it will be classified as successful fix or unsuccessful. Based on the classification, an appropriate response prompt is going to be sent to ChatGPT.

To achieve such objectives. different prompt type and strategies will be tried to get the best performance out of the NLP model.

HPC Performance Bug List

1) Inefficient coding for target micro-architecture
    1.1) Memory/Data locality
    1.2) Micro-architectural inefficiency

2) Missing parallelism
    2.1) Vector/SIMD parallelism
    2.2) GPU parallelism
    2.3) Instruction level parallelism
    2.4) Task parallelism

3) Parallelization overhead/inefficiency
    3.1) Small parallel region
    3.2) Inefficient thread mapping / Inefficient block size / Load imbalance
    3.3) Under-parallelization
    3.4) Over-Parallelization

4) Inefficient Concurrency control and synchronization
    4.1) Unnecessary locks
    4.2) Unnecessary strong memory consistency
    4.3) Lock management overhead
    4.4) Unnecessary synchronization

5) Unnecessary process communication

6) Inefficient algorithm /data-structure and their implementation
    6.1) Unnecessary operation/traversal/function call
    6.2) Redundant operation
    6.3) Expensive operation
    6.4) Frequent function call
    6.5) Inefficient data-structure library
    6.6) Usage of improper data type

7) Inefficient memory management
    7.1) Memory leak
    7.2) Repeated memory allocation
    7.3) Redundant memory allocation
    7.4) Slower memory allocation library call
    7.5) Insufficient memory
    7.6) Unnecessary data copy

8) I/O inefficiency
    8.1) Sequential I/O operation
    8.2) Over parallelization

9) Unintentional Programming logic error

10) Inefficiency due to new compiler version