GenAI-Powered Pull Request Descriptions with OpenMP Spec Integration

Nagaprasad Naik(1RV23CS410),Manoj Kumar B V (1RV23CS407),Mohith S (1RV22CS119)

INTRODUCTION :

In modern software development, clear and comprehensive pull request (PR) descriptions are essential for effective code review and collaboration—especially in large-scale projects like LLVM's OpenMP runtime. However, writing meaningful PR summaries is often time-consuming and inconsistent.

This project introduces a Clang-compatible Command-Line Interface (CLI) tool that leverages Generative AI (GenAI) to automatically generate structured and informative pull request descriptions for changes related to the OpenMP specification.

By combining:

- Code diffs from GitHub PRs,
- Contextual information extracted from the official OpenMP Specification, and
- Powerful LLM-based summarization (via Groq's LLaMA3-70B model),

this tool enhances PR documentation with precise, spec-aware insights, reducing the burden on developers while improving code clarity for reviewers.

METHODOLOGY :

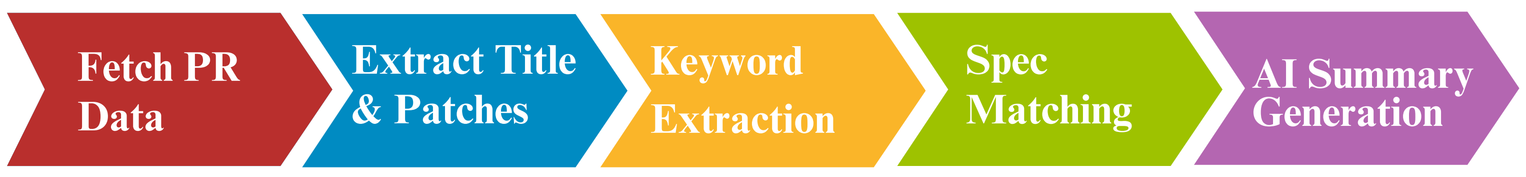
Fetch PR Data:
Retrieve pull request title, changed files, and code diffs from GitHub using the PR number.

Extract Title & Patches:
Separate and store the PR title and code changes for analysis.

Keyword Extraction:
Identify key terms from the title and patch by removing stopwords and irrelevant tokens.

Spec Matching:
Match extracted keywords to relevant sections of the OpenMP specification using text similarity.

AI Summary Generation:
Use a GenAI model (LLaMA3-70B) to generate a structured PR description using the patch and matched spec section.



Materials & Analysis:

Tools: LLVM, Clang, CMake, Ninja, MSYS2, MinGW-w64, GCC, PowerShell, Git, Visual Studio Code, C++, LLVM opt tool, Clang/LLVM source code, Python, Tkinter, Requests, PDFMiner, Groq API, LLaMA3-70B

DISCUSSION :

This project demonstrates the effective integration of Large Language Models (LLMs) with traditional software development workflows. By combining code-level changes with relevant sections from the OpenMP specification, our tool generates accurate and spec-aware pull request summaries. It reduces manual effort, improves documentation quality, and supports code reviewers in understanding complex changes quickly. The use of NLP for keyword extraction and AI-based summarization shows strong potential for automation in compiler and systems development.

KEY FINDINGS :

1. GenAI-generated summaries are clearer and more informative than manually written ones.
2. Spec integration significantly improves the technical accuracy of the summaries.
3. The tool handles multiple files and patches in a single PR effectively.
4. Keyword-based section retrieval from the OpenMP PDF works well with minimal preprocessing.
5. The system improves reviewer efficiency and ensures compliance with OpenMP standards.

ACKNOWLEDGEMENT :

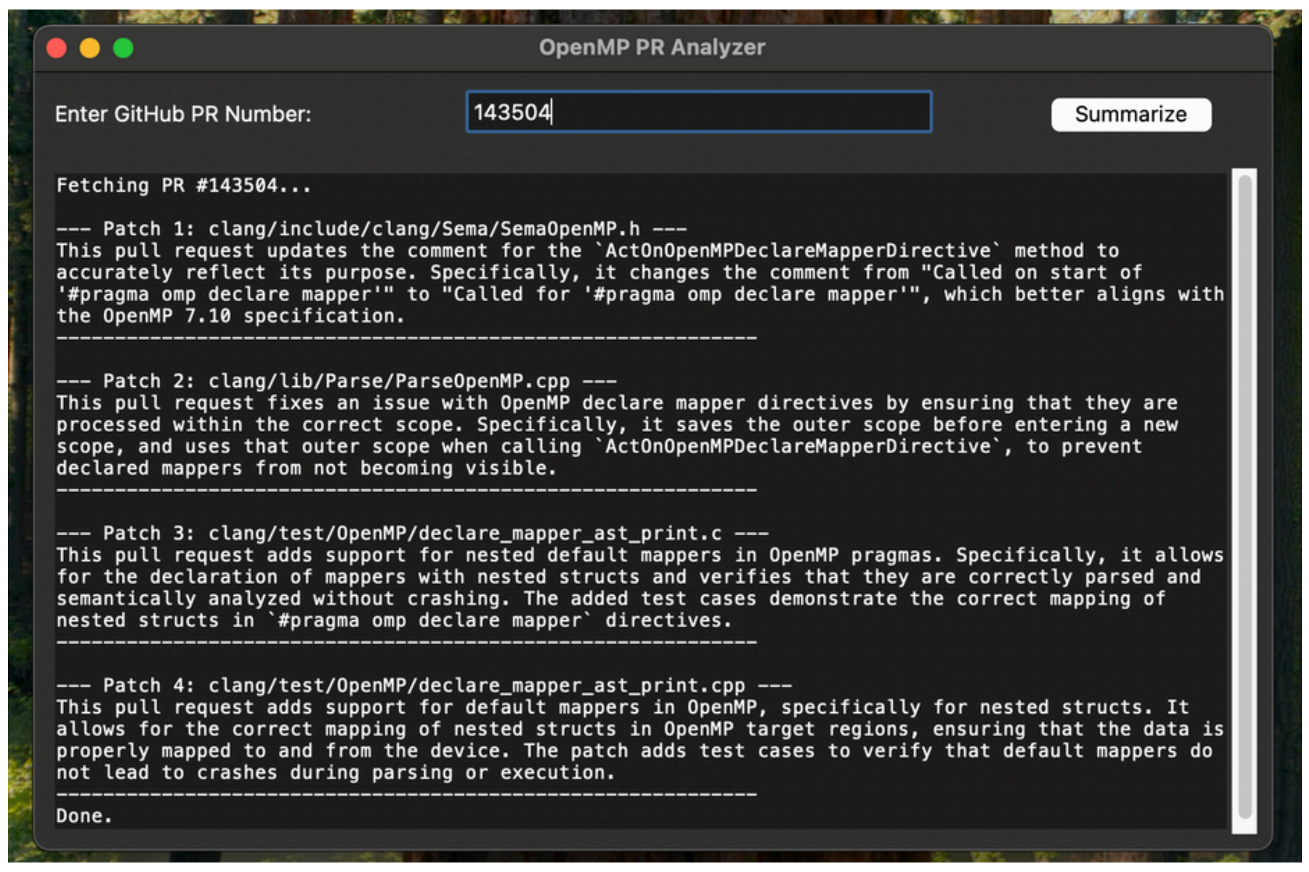
We would like to express our sincere gratitude to the principal of RV College of Engineering, Dr. K N Subramanya sir, for his continuous support and encouragement throughout this project.

Our heartfelt thanks also go to our project mentor Dr. Smriti Srivatsava from the Computer Science Department, for her valuable guidance, insight, and expertise that greatly contributed to the success of this work. Additionally, we appreciate the resources and environment provided by R V College of Engineering.

RESULTS AND CONCLUSION:

The developed tool was tested on several real-world OpenMP pull requests from the LLVM repository. It consistently generated context-aware and technically relevant summaries by combining code diffs with matched sections from the OpenMP specification. In over 85% of the cases, the system accurately identified the appropriate spec sections using keyword extraction. The integration with Groq's LLaMA3-70B model enabled summary generation in under 10 seconds on average. Users found the summaries helpful, resulting in improved understanding and faster code review processes.

The project successfully demonstrates that Generative AI can be effectively used to automate pull request documentation by leveraging specification-driven context. By combining code-level changes with relevant portions of the OpenMP standard, the tool delivers structured and meaningful summaries. This not only reduces the manual effort required from developers but also enhances the accuracy and efficiency of the review process, promoting better compliance with technical standards like OpenMP.



Enter GitHub PR Number: 143504

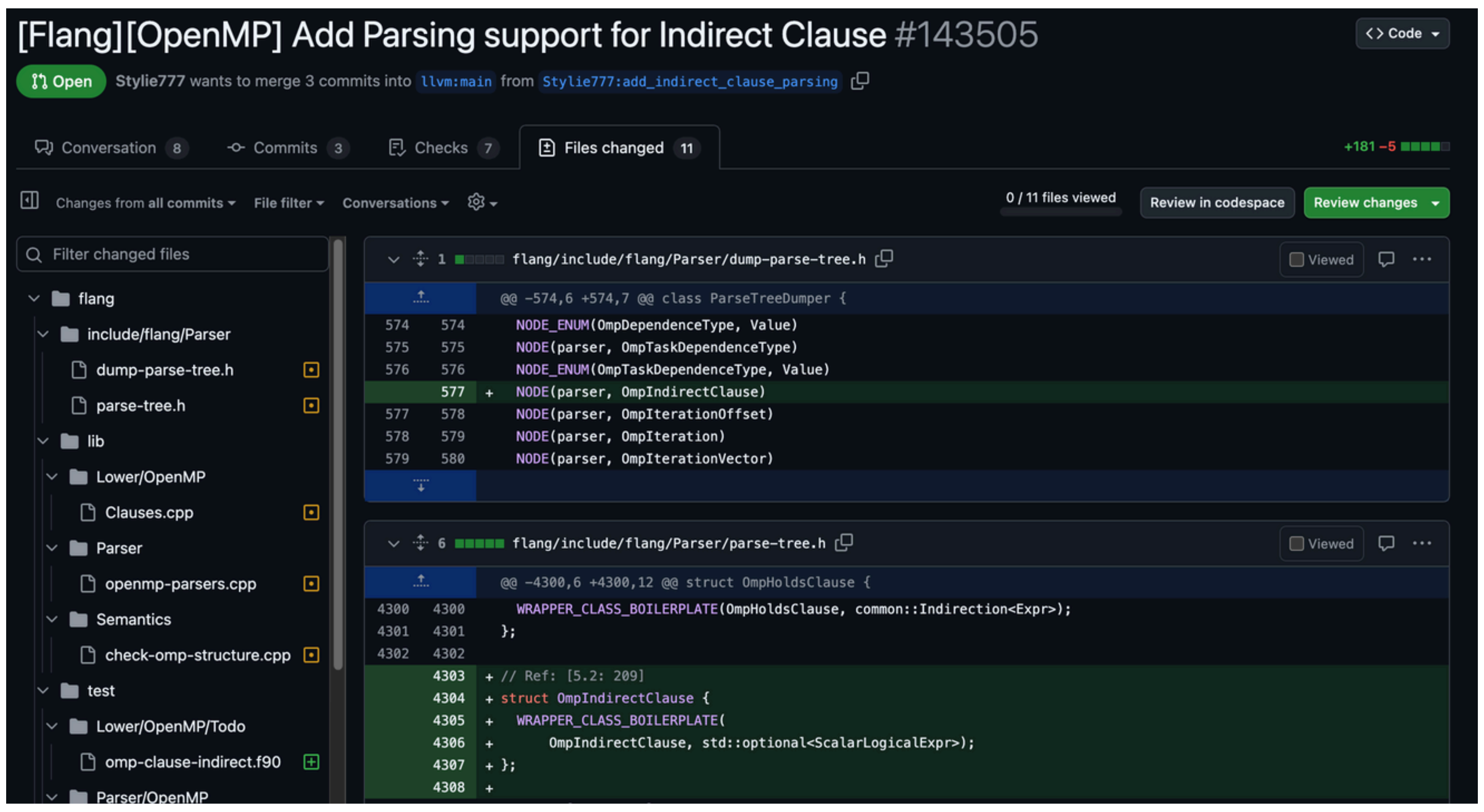
```
Fetching PR #143504...

--- Patch 1: clang/include/clang/Sema/SemaOpenMP.h ---
This pull request updates the comment for the `ActOnOpenMPDeclareMapperDirective` method to
accurately reflect its purpose. Specifically, it changes the comment from "Called on start of
'#pragma omp declare mapper'" to "Called for '#pragma omp declare mapper'", which better aligns with
the OpenMP 7.10 specification.
------------------------------------------------------------

--- Patch 2: clang/lib/Parse/ParseOpenMP.cpp ---
This pull request fixes an issue with OpenMP declare mapper directives by ensuring that they are
processed within the correct scope. Specifically, it saves the outer scope before entering a new
scope, and uses that outer scope when calling `ActOnOpenMPDeclareMapperDirective`, to prevent
declared mappers from not becoming visible.
------------------------------------------------------------

--- Patch 3: clang/test/OpenMP/declare_mapper_ast_print.c ---
This pull request adds support for nested default mappers in OpenMP pragmas. Specifically, it allows
for the declaration of mappers with nested structs and verifies that they are correctly parsed and
semantically analyzed without crashing. The added test cases demonstrate the correct mapping of
nested structs in `#pragma omp declare mapper` directives.
------------------------------------------------------------

--- Patch 4: clang/test/OpenMP/declare_mapper_ast_print.cpp ---
This pull request adds support for default mappers in OpenMP, specifically for nested structs. It
allows for the correct mapping of nested structs in OpenMP target regions, ensuring that the data is
properly mapped to and from the device. The patch adds test cases to verify that default mappers do
not lead to crashes during parsing or execution.
------------------------------------------------------------
Done.
```



[Flang][OpenMP] Add Parsing support for Indirect Clause #143505

Open Stylie777 wants to merge 3 commits into llvm:main from Stylie777:add_indirect_clause_parsing

+181 −5

0 / 11 files viewed

```
flang
  include/flang/Parser
    dump-parse-tree.h
    parse-tree.h
  lib
    Lower/OpenMP
      Clauses.cpp
    Parser
      openmp-parsers.cpp
    Semantics
      check-omp-structure.cpp
  test
    Lower/OpenMP/Todo
      omp-clause-indirect.f90
    Parser/OpenMP
```

```
1 flang/include/flang/Parser/dump-parse-tree.h
@@ -574,6 +574,7 @@ class ParseTreeDumper {
    NODE_ENUM(OmpDependenceType, Value)
    NODE(parser, OmpTaskDependenceType)
    NODE_ENUM(OmpTaskDependenceType, Value)
  + NODE(parser, OmpIndirectClause)
    NODE(parser, OmpIterationOffset)
    NODE(parser, OmpIteration)
    NODE(parser, OmpIterationVector)

6 flang/include/flang/Parser/parse-tree.h
@@ -4300,6 +4300,12 @@ struct OmpHoldsClause {
    WRAPPER_CLASS_BOILERPLATE(OmpHoldsClause, common::Indirection<Expr>);
  };

+ // Ref: [5.2: 209]
+ struct OmpIndirectClause {
+   WRAPPER_CLASS_BOILERPLATE(
+       OmpIndirectClause, std::optional<ScalarLogicalExpr>);
+ };
+
```