LLVM GsoC 2018 Proposal

A single updater class for Dominators in LLVM

PERSONAL INFORMATION

Name	Sagar Thakur
Country	India (GMT + 5:30)
School	International Information of Information Technology, Hyderabad https://www.iiit.ac.in/about/quick-facts/
Degree	Master of Technology in Computer Science and Engineering
Email	cs.sagarthakur@gmail.com sagar.thakur@students.iiit.ac.in
Contact	(+91) 8446605703

RELEVANT EXPERIENCE

Before joining IIIT hyderabad I was working at Imagination technologies where I gained 3 years of experience of working in the LLVM community. I am well aware of the source code structure and the building process. I have worked on the LLVM, LLDB and the compiler-rt projects of LLVM. My work was mainly focused on porting LLVM, LLDB and compiler-rt to support the MIPS backend. My patches and commits can be found at https://reviews.llvm.org/p/slthakur/ . I also have experience working with the LLVM IR. My work on LLVM IR was mainly focused on lowering the IR to mips target assembly in the chromium subzero project. I have also worked with selection DAGs in LLVM. My patches in the subzero project can be found at:

https://codereview.chromium.org/search?closed=1&owner=sagar.thakur&reviewer=&cc=&repo_guid=&base=&project=&private=1&commit=1&created_before=&created_after=&modified_before=&modified_after=&order=&format=html&keys_only=False&with_messages=False&cursor=&limit=30

I have gone through the relevant research papers mentioned in the comments in the source code and the dominator tree class reference diagrams. I have run the dominator tree IR tests using debug flags for the dominator tree to see how it is built.

I have studied graduate level compiler construction and optimization courses. I had also taken up a course on Advanced Problem Solving which teaches how to model data structures to solve complex computer science problems. In the Advanced problem solving course I have acquired knowledge of basic and advanced graph and tree algorithms. I have also worked on implementation of Tarjan's incremental and decremental dynamic depth first search algorithm as a project in the Advanced Problem Solving course. Apart from this I have worked on various Operating System/ Database System/Scripting mini projects.

OBJECTIVE OF THE PROJECT

Design and implement a new class for abstracting away how the dominator tree updates are performed when the CFG changes.

CRITERIA OF SUCCESS

1. A clean abstract updater interface for the dominator tree and the post dominator tree for the users of the dominator tree.
2. Faster incremental updates to the dominator tree and the post dominator tree. Using the new class we would be able to avoid the unnecessary changes to the Post dominator tree

MOTIVATION

Although I have worked previously in llvm, my work was mainly focused on the porting of MIPS processor for various LLVM projects. This is the first time I will be working on the generic part of LLVM. The area of dominator trees and such compiler optimizations is completely new to me and I always wanted to work on the internal data structures used in compiler optimization.I believe this project will be a good start for me to get into the field of compiler optimization.

PROPOSAL

Going through the source code I have tried to understand where the changes need to be made. Here is the plan I propose to follow based on my understanding of the source code:

- Wrap all the dominator tree related classes such as the DominatorTree class, DominatorTreeAnalysis class and DefferedDominance class defined in Dominators.h with a single updater class.
- Make code changes in the places that use the dominator tree such as the LoopSimplify, BreakCriticalEdges, LoopUnroll etc. Now we will use the single updater class.
- To make the updation faster I will design an algorithm to prune any unnecessary changes to the Post Domination Tree.
- Design new strategies to perform lazy updates.

TIMELINE

I have tried to come up with a preliminary timeline which could be followed to complete the project. I am open to any change in the timeline according to the mentor.

Duration	Description
April 23	Acceptance of students proposals
April 23 to April 30 (Community Bonding Period)	• Initial discussion about requirements for various modules with mentor. • Set up of weekly schedule for updates via skype/email/chat. • Explore source code with current state of repository.
April 30 to May 7 (Community Bonding Period)	• Discuss requirements with mentor. • Continue understanding current codebase. • Start designing of the inheritance diagram and collaboration diagram of the updater class.

May 7 –to May 14 (Community Bonding Period)	• Finalize implementation details. • Start implementation of the updater class.
May 14 to May 21	• Finish implementation of the updater class and submit patch for it. • Address the review comments on the patch and commit the changes
May 21 to May 28	• Use the new updater object to the existing code which was working directly with the dominator / deferred dominator tree. • Write and evaluate test cases • Submit patch the changes and the test cases
May 29 to June 10	• Address the review comments on the patch. • Add more tests if needed. • Commit the changes after acceptance of the patch.
June 11 to June 15 (Phase 1 Evaluation)	• Add documentation if needed. • Discuss progress with mentor.
June 16 to June 24	• Start design and visualization of algorithm to prune unnecessary PostDomTree updates based on updates to the DomTree. • Finalize design of algorithm with mentor.
June 25 to July 2	• Implement the algorithm. • Write and evaluate test cases. • Submit patch for it • Address review comments and commit patch after acceptance
July 2 to July 9	• Compare performance metrics for previous implementation and current implementation
July 9 to July 13 (Phase 2 Evaluation)	• Add documentation if needed. • Discuss progress with mentor.

July 14 to Aug 5	● Explore more strategies for performing lazy updates. ● Discuss with mentor about various strategies. ● Implement the best strategy discussed with mentor. ● Write and evaluate test cases. ● Submit patch and commit on acceptance. ● Complete any unfinished task, take up some new tasks that require to be implemented as per the mentor's directive
Aug 6 to Aug 14	● Final week: Submit final product and wait for mentor's evaluation

AVAILABILITY

My semester exams finish by the 26th of April allowing me to get started right away after that. I have no prior commitments during the months of May, June and July. College reopens in the 1st week of August, which may slightly affect my schedule with respect to timings, but I intend to finish a major portion of the work during the summer itself. I would be able to work 8+ hours on a daily basis. Thus I will be able to devote a minimum of 56 hours per week for the project.

A NOTE OF THANKS

I would like to thank the llvm community for giving me this opportunity to write a proposal. I look forward to receiving feedback from those reviewing this document, and would be glad to discuss/change accordingly.