Utilize LoopNest Pass

Toshiki Maekawa
April 14th 2021

1. Abstract

A newly added idea of LoopNest pass enables us to handle a loop nest efficiently, unlike function pass or loop pass. The goals of this project are to utilize LoopNest pass for some existing analyses/transformations intended to run on loop nest, collect data on the utilized ones (e.g. compiling time), and improve the current implementation of LoopNest pass itself as necessary.

2. Introduction and Goals

As mentioned in Abstract, the goals of this project are:

- Utilize LoopNest pass for some existing analyses/transformations
- Collect performance data on utilized ones (e.g. compiling time)
- Improve the current implementation of LoopNest pass

Some analyses/transformations are intended to run on a loop nest or potentially handle a loop nest, but are currently implemented as function pass or loop pass. By utilizing LoopNest pass for those ones, we can naturally perform analyses/transformations on a loop nest and can expect side benefits.

In this project, at least these 5 passes will be made changes:
LoopUnrollAndJamPass, LoopFusionPass, LoopCachePrinterPass(CacheCost), **LoopNestPrinterPass**, **and LoopIdiomRecognizePass**.
I found it hard to determine if an analysis/transformation is good for this project before coding and further discussion starts. Thus, we will be finding more candidates through the project, while making changes for the passes mentioned above.
In addition, some analyses (or *utilities for analysis*) are not implemented as passes. For such cases, we simply utilize LoopNest as a primary object.

3. Implementation Plan

3.1 Finding good passes for the change

Needless to say, in this project, some passes will be changed to LoopNest passes. However, this change is not good for particular passes. For example, loop passes used in a loop pipeline may be not suitable because the order of processing loops varies due to the change.

Currently, these loop passes are used in a loop pipeline: **LoopInstSimplifyPass, LoopSimplifyCFGPass, LoopRotatePass, LICMPass, SimpleLoopUnswitchPass, LoopIdiomRecognizePass, IndVarSimplifyPass, LoopDeletionPass, LoopInterchangePass** (utilized LoopNest by [1])**, LoopFullUnrollPass.**
I had better avoid these passes in the early part of this project, so I can avoid troublesome problems caused by unfamiliarity with the passes.

Considering the problem mentioned above, I chose 5 passes for this project: **LoopUnrollAndJamPass, LoopFlattenPass, LoopCachePrinterPass/CacheCost, LoopNestPrinterPass,** and **LoopIdiomRecognizePass (challenging though).**
While utilizing LoopNest for these passes, more analyses/transformations will be discussed to be made changes. Actual changes will be made in the latter part of this project.

3.2 Timeline

Week:

1. **Prior – May 17** Work on small patches ([2] in progress)
2. **May 18 – May 24** Start working on LoopUnrollAndJamPass
   - Use LoopNest as a primary object
   - Support loop more than 2 levels if possible
3. **May 25 – June 31** Finish working on LoopUnrollAndJamPass
   Run benchmarks and collect data
4. **June 1 – June 7** Start working on LoopFlattenPass
   - Use LoopNest as a primary object
   - Currently, flattening processes a pair of loops (outer, inner). This can be faster if processing all loops in a loop nest at once.
5. **June 8 – June 14** Finish working on LoopFlattenPass
   Run benchmarks and collect data
6. **June 15 – June 21** Work on LoopCachePrinterPass (i.e. CacheCost)

- This pass only runs at the top of a loop nest, so it's inefficient to be implemented as a loop pass.
- Internally calls CacheCost::getCacheCost. This is a good practice to utilize LoopNest for utility for analysis.

Run benchmarks and collect data

7. **June 22 – June 28** Work on LoopNestPrinterPass

   Start working on LoopIdiomRecognizePass
   - Add idioms for more than one loop … and so on
8. **June 29 – July 5** Finish working on LoopIdiomRecognizePass

   Run benchmarks and collect data

   Discuss if LoopIdiomRecognizePass is worth to implement as a LoopNest pass
9. **July 6 – July 12** Work on a pass found good for LoopNest during the project
10. **July 13 – July 19** Prepare for evaluation
11. **July 20 – July 26** Work on an analysis found good for LoopNest during the project
12. **July 27 – Aug 2** Improve the current implementation of LoopNest pass
13. **Aug 3 – Aug 9** Fix bugs emerged during the project
14. **Aug 10 – Aug 17** Prepare for final evaluation

4. Bio

- Name: Toshiki Maekawa
- Country: Japan
- Timezone: UTC+9 (Japan Standard Time)
- Email: [redacted]
- Affiliation: 2nd-year undergraduate student at Nagoya Institute of Technology, majoring in computer science
- GitHub: https://github.com/maekawatoshiki
- I have no difficulty in text communication in English.

4.1 Availability

According to the changes [3], we students will be focused on a 175-hour project over a 10-week coding period in 2021. This change encouraged me to participate in GSoC. I can afford to work on GSoC for 24 hours at least per week. However, my college holds lectures until mid-August, so it'll be a bit difficult to spend time on coding during exam week.

4.2 Motivation

I got interested in compilers when I was 13 years old. Since then, I'm really into it and have made many compilers ({C, Subset of OCaml, my own languages} compiler), interpreters ({Subset of Ruby, JavaScript, my own languages} interpreter), virtual machines (e.g. JVM, CLI/ecma335) and things related to them (HTML/CSS rendering engine from scratch). (Most of them are public on my GitHub)

LLVM was the first compiler infrastructure I met. I clearly remember that moment I was impressed with the idea of the project. I've used LLVM many times for my personal projects, and I thought now is the time to contribute to LLVM.

I've been interested in speeding up the compiler optimization process. I didn't know about LoopNest pass before knowing the project idea, but I thought it a great opportunity to become more familiar with improving the performance of compiler optimization.

5. References

[1] https://reviews.llvm.org/D97847

[2] https://reviews.llvm.org/D99149

[3] https://opensource.googleblog.com/2020/10/google-summer-of-code-2021-is-bringing.html