

Improve inter-procedural analyses and optimizations

GSoC 2021 Final Report

Student: Kuter Dinel - kuterdinel@gmail.com

Overview

In this project I worked towards improving LLVM's inter procedural optimization capabilities. Mainly focusing around the Attributor framework.

Work done

Call edges, optimistic call graph

I added an AbstractAttribute that computes the optimistic live call edges using the attributor liveness information.

Using the custom iterator implemented on the GraphTraits interface you can use it as a `optimistic` call graph. It is also possible to print it as a dot graph.

This is used by the inter-procedural function reachability attribute.

<https://reviews.llvm.org/D104059>

Function reachability.

I added an AbstractAttribute that uses Attributor's internal 'optimistic' call graph that I wrote in a previous patch to answer queries about function call reachability. Functions can become reachable over time as new call edges are discovered.

<https://reviews.llvm.org/D104599>

Deducing AMDGPU Attributes with the attributor.

AMDGPU backend has some target specific(string) attributes that indicate if certain features are used by a function, for example if a function calls `llvm.amdgcn.workitem.id.x` intrinsic, or calls a function that calls this intrinsic. This information is used during code generation and has to be deduced through inter-procedural analysis. I wrote a patch that uses the Attributor framework to deduce these attributes.

<https://reviews.llvm.org/D104997>

Propagating HotCold Inter-procedurally

Propagate Cold inter-procedurally

This is a simple patch that adds basic propagation of cold attribute across functions. It assumes that a function is cold if all of its callers are cold also.

<https://reviews.llvm.org/D107409>

Propagate Hot/Cold to call sites in the same block.

This is the second patch in the patchset. It uses the other calls inside the same block to infer if a call site is cold or hot. We assume that a call site is considered hot if there is a call to a hot inside the same block. If all call sites of a function is hot or cold we consider the function hot or cold respectively.

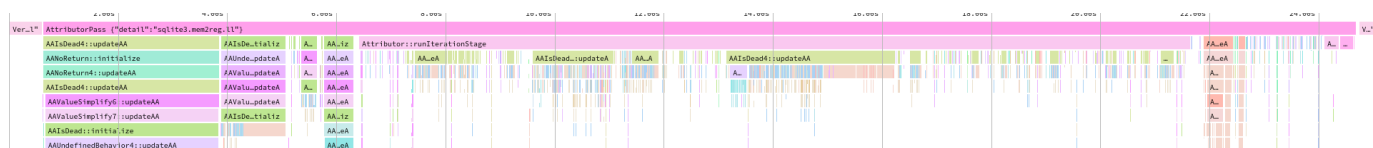
Use CFG to propagate hot/cold.

This patch makes it possible to propagate hot/cold across basic blocks within the attributes. It relies on the BranchProbabilityInfo to check if an edge is hot or cold. The hot cold information is propagated.

<https://reviews.llvm.org/D108008>

Splitting Attributor to multiple stages

We believe that it would improve performance to split attributor into multiple stages. Running Liveness and hot cold propagation before anything else. This way we wouldn't create attributes in positions that are considered dead, also we can use hot/cold information to decide whether we want to seed an Attribute or not, This would allow us to limit the compile time that we spend on optimizing cold code.



(Flame graph of the attributor running in 2 stages)

<https://reviews.llvm.org/D108099>

Issues fixed in the Attributor

checkForAllInstructions correctly handle declared functions.

checkForAllInstructions was not correctly handling the case when the function was a declaration (functions that don't have a body).

<https://reviews.llvm.org/D106625>

AAPointerInfo uninitialized usage.

AAPointerInfo was using an uninitialized value inside a branch.

<https://reviews.llvm.org/D108170>

Conclusion

Just like last year I really enjoyed working on interprocedural optimizations and I have learned a lot and gained a lot of experience.

I will continue contributing to the LLVM and the Attributor framework in my free time.