Modern compilers attempt to optimize programs as much as possible. One such significant effort is Link–Time Optimization (LTO). LTO takes the whole program as accessible to a linker, and performs global optimizations that are impossible in previously local compilations. Because of the global nature, LTO must be performed in full in each compilation, which results in long compile times even in edit–compile cycles. Incremental compilation can reduce compile times of edit–compile cycles by reusing unchanged objects.

This thesis aims to implement incremental compilation for Link–Time Optimizations of GNU Compiler Collection, specifically of local transformation phase. We implement incremental compilation by caching files of compilation units of local transformation.

For best success of incremental compilation we also aim to minimize number of changed compilation units after small edit. We achieve this in two ways. First, we create better partitioning strategy, that concentrates the changes into fewer compilation units. Second, we analyzed sources of divergence and, if easily possible, removed them. That includes stabilizing values and fixing streaming and inter–procedural optimizations to increase their robustness against small edits. Both without influencing quality of the final binary.