LTO BOF

Honza Hubička

Institute of Computer Science
Charles University
Prague

Mathematics and Statistics
University of Calgary
Calgary

GNU Cauldron 2015

LTO in GCC 5

- Streaming of command line options
  - GCC now handle well many cases where different command line options are used at compile time
  - Infrastructure based on `optimize` and `attributes`
  - We now need to take those seriously: many options in `*.opt` are not annotated correctly or not implemented in context sensitive manner.

LTO in GCC 5

- Streaming of command line options
  - GCC now handle well many cases where different command line options are used at compile time
  - Infrastructure based on `optimize` and `attributes`
  - We now need to take those seriously: many options in `*.opt` are not annotated correctly or not implemented in context sensitive manner.
- OpenMP offloading infrastructure
  - For the first time we want to use LTO streaming for other purposes
  - IL is streamed by GCC with one backend and read by GCC with different one

LTO in GCC 5

- Streaming of command line options
  - GCC now handle well many cases where different command line options are used at compile time
  - Infrastructure based on `optimize` and `attributes`
  - We now need to take those seriously: many options in `*.opt` are not annotated correctly or not implemented in context sensitive manner.
- OpenMP offloading infrastructure
  - For the first time we want to use LTO streaming for other purposes
  - IL is streamed by GCC with one backend and read by GCC with different one
- memory use/streaming is about 8% down
  - A lot of small changes to avoid unnecessary data in global data-structures
  - Better early code removal

LTO in GCC 5

- Streaming of command line options
  - GCC now handle well many cases where different command line options are used at compile time
  - Infrastructure based on `optimize` and `attributes`
  - We now need to take those seriously: many options in `*.opt` are not annotated correctly or not implemented in context sensitive manner.
- OpenMP offloading infrastructure
  - For the first time we want to use LTO streaming for other purposes
  - IL is streamed by GCC with one backend and read by GCC with different one
- memory use/streaming is about 8% down
  - A lot of small changes to avoid unnecessary data in global data-structures
  - Better early code removal
- Code quality improvements

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)
- Better devirtualization: dynamic type detection (14% fewer virtual calls remains)

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)
- Better devirtualization: dynamic type detection (14% fewer virtual calls remains)
- IPA-CP reorg separating type propagation from constant propagation; alignment propagation (182% more devirts, 175% increase in number of cloned bodies)

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)
- Better devirtualization: dynamic type detection (14% fewer virtual calls remains)
- IPA-CP reorg separating type propagation from constant propagation; alignment propagation (182% more devirts, 175% increase in number of cloned bodies)
- Better early unreachable code removal (10% fewer symbols enter LTO)

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)
- Better devirtualization: dynamic type detection (14% fewer virtual calls remains)
- IPA-CP reorg separating type propagation from constant propagation; alignment propagation (182% more devirts, 175% increase in number of cloned bodies)
- Better early unreachable code removal (10% fewer symbols enter LTO)
- Inliner now leads to less of code bloat with LTO (10% smaller binary non-FDO, 14% with FDO)

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)
- Better devirtualization: dynamic type detection (14% fewer virtual calls remains)
- IPA-CP reorg separating type propagation from constant propagation; alignment propagation (182% more devirts, 175% increase in number of cloned bodies)
- Better early unreachable code removal (10% fewer symbols enter LTO)
- Inliner now leads to less of code bloat with LTO (10% smaller binary non-FDO, 14% with FDO)
- Comdat localization

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)
- Better devirtualization: dynamic type detection (14% fewer virtual calls remains)
- IPA-CP reorg separating type propagation from constant propagation; alignment propagation (182% more devirts, 175% increase in number of cloned bodies)
- Better early unreachable code removal (10% fewer symbols enter LTO)
- Inliner now leads to less of code bloat with LTO (10% smaller binary non-FDO, 14% with FDO)
- Comdat localization
- Write only variables removal

All code quality changes are tested at Firefox

- Identical code folding (10% of functions are merged; still not as effective as gold's aggressive ICF)
- Better devirtualization: dynamic type detection (14% fewer virtual calls remains)
- IPA-CP reorg separating type propagation from constant propagation; alignment propagation (182% more devirts, 175% increase in number of cloned bodies)
- Better early unreachable code removal (10% fewer symbols enter LTO)
- Inliner now leads to less of code bloat with LTO (10% smaller binary non-FDO, 14% with FDO)
- Comdat localization
- Write only variables removal
- Stronger privatization

LTO in GCC 6

- Early debug
  - Dwarf2out implementation merged to trunk by Aldy in June
  - LTO part prototyped by Richard
- Alias analysis
  - LTO TBAA seems a lot of fun! Will tell more tomorrow :)
  - Main issue is how to merge types to comply with all language standards and common practices
- IPA-ICF improvements
  - Gimple operand matching can partly merge with fold-const?
  - Tail merging and IPA-ICF can merge
- Early optimization improvements
  - Is there any low hanging fruit?
- More tuning on non-x86_64 architectures.
- Finish command line options handing
- Can we skip GAS when producing slim LTO objects?
- More plans?

Thank you!