Gold

Ian Lance Taylor
Google

April 16, 2010

What?

What is gold?

- gold is a new linker, first released March 2008.
- gold is now part of the GNU binutils (if you configure with `--enable-gold`, gold is built instead of GNU ld).
- gold only supports ELF, which is used by all modern operating systems other than Mac OS and Windows.
- gold is written in C++.
- gold currently supports x86, x86_64, ARM, and SPARC.

Why?

Why write a new linker?

- Almost all programmers use no linker features.
  - Exception: Linux kernel build
  - Exception: linker scripts on embedded systems
  - Exception: version scripts for libraries
- The linker is a speedbump in the development cycle.
- Compilation can be easily distributed; linking can not.
- The GNU linker is slow.

Why?

Why is the GNU linker slow?

- It was designed for the a.out and COFF object file formats. ELF support was added later.
- ELF includes relocations which build new data; this had to be shoehorned into the GNU linker.
- The GNU linker traverses the symbol table thirteen times in a typical link.
  - gold traverses the symbol table three times.
- The GNU linker is built on top of BFD, increasing the size of basic data structures like symbol table entries.
  - For x86_64, GNU linker symbol table entry is 156 bytes.
  - gold is 68 bytes.
- The GNU linker always loads values using byte loads and shifts.

Why?

Why not fix the GNU linker?

- The GNU linker source code is split in several parts which communicate by various hooks.
  - The linker proper (`src/ld`).
  - The ELF emulation layer (`src/ld/emultempl/elf32.em`).
  - The generic BFD library (`src/bfd`).
  - The ELF support in the BFD library (`src/elf.c`, `src/elflink.c`).
  - The processor specific ELF backend (e.g., `src/elf64-x86-64.c`).
- The GNU linker is designed around a linker script. All actions are driven by entries in the linker script.

Why?

Why not fix the GNU linker?

- The GNU linker source code is split in several parts which communicate by various hooks.
  - The linker proper (`src/ld`).
  - The ELF emulation layer (`src/ld/emultempl/elf32.em`).
  - The generic BFD library (`src/bfd`).
  - The ELF support in the BFD library (`src/elf.c`, `src/elflink.c`).
  - The processor specific ELF backend (e.g., `src/elf64-x86-64.c`).
- The GNU linker is designed around a linker script. All actions are driven by entries in the linker script.

Changing this design is not a fix; it is a rewrite.

How?

Some notes on the gold implementation. For details, see the source code.

- Over 90,000 lines of commented C++ code.
- Uses templates to avoid byte swapping for a native link.
- Multi-threaded.
- Not driven by a linker script.
  - Linker scripts are supported, though.
  - Linker script support is over 10% of the target independent source code.
- Includes plugin support, used by gcc and LLVM for whole program optimization.

Performance

How long it takes gold to link compared to the GNU linker.

- Hello, world
  - Dynamic link: 37% faster
  - Static link: 54% faster
- Large program (700M, 1300 objects, 400,000 symbols)
  - Complete build from scratch: 50% faster
  - Change one input object: 82% faster
  - Difference is disk cache effects.

gold and Linux

- Using gold to build the Linux kernel breaks regularly.
- The Linux kernel build uses complex linker scripts.
- Worse, the Linux kernel build uses undocumented features of complex linker scripts.
- As these features are discovered, they are added to gold.

gold Linux features

Features added to gold solely for the Linux kernel build:

- Linking binary non-ELF input files.
- Generating binary non-ELF output file.
- The `--emit-relocs` option.

gold script incompatibilities

Undocumented GNU ld features where gold had to be fixed to be compatible solely for the Linux kernel build:

- Alignment of data section in a binary input file.
- Placement of file header when linker script explicitly lists program segments.
- Merging sections with the same name but different flags in some cases but not others.
- Handling of orphan `.eh_frame` sections in linker script.
- Aliases for symbols defined in linker script.
- Do not warn about references from a debugging section to a symbol in a section which is discarded by a linker script.
- Do not warn about redefining a symbol first seen in an object used with `--just-symbols`.
- Permit ordinary objects within `--start-group` `--end-group`.

How to help

How can Linux developers help gold?

Please test linker script changes with both GNU ld and gold.

This will help make it possible for gold to become the default linker on Linux for supported architectures. That is good for developers working on Linux, because gold is faster.

(The other main thing which needs to be addressed before gold can become the default is the glibc build process.)

How gold can help

How can gold help Linux developers?

gold is faster but linking time is not a big obstacle for a typical kernel build. Also gold is not that much faster when using a linker script.

Use thin archives (supported by `ar` as of binutils 2.19) and `--whole-archive` instead of `ld -r`. Thin archives are more efficient because they do not require copying the object files. They were developed for gold but also work with GNU ld.

Incremental Linking

Problem: changing one object file only changes a small part of an executable. Recreating the entire executable is wasteful.

- Solution: incremental linking.
- The linker records symbol and relocation information in the executable.
- The linker checks which objects are newer than the executable.
- Only those objects are updated.
- If only object changes, there is significantly less relocation processing and significantly less I/O.
- This has been designed in some detail, and work is starting on an implementation.

Who

- Ian Lance Taylor
  - Design, bulk of implementation.
- Cary Coutant
  - Shared library generation, TLS, plugins.
- Sriraman Tallam
  - Garbage collection, identical code folding.
- Craig Silverstein
  - x86_64 port, ODR detection, debug info compression.
- Andrew Chatham
  - x86_64 port.
- David Miller
  - SPARC port.
- Doug Kwan
  - ARM port, relaxation support.
- Viktor Kutuzov
  - ARM port.
- Kris Van Hees
  - Key patches for Linux build.