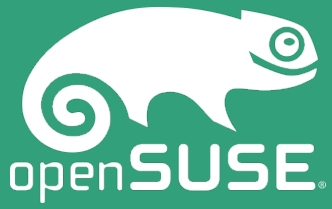


Building openSUSE with link-time optimizations

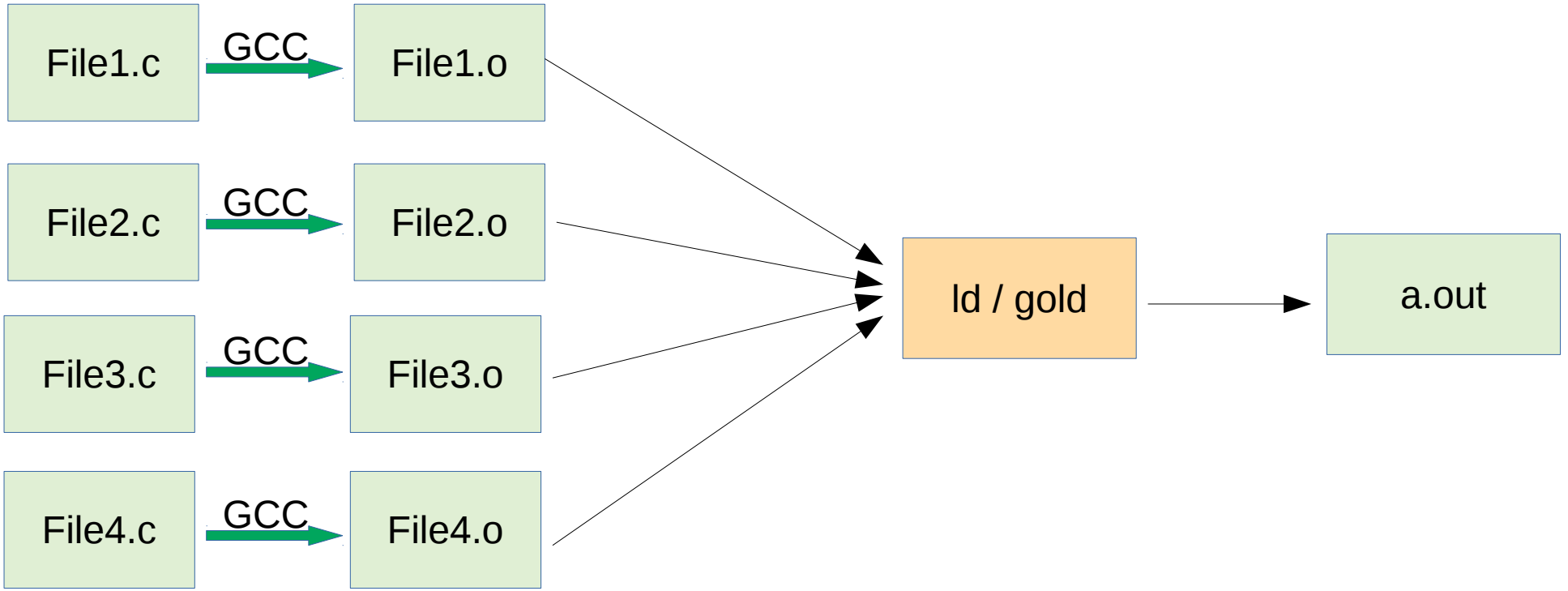
Jan Hubička and Martin Liška
SUSElabs
jh@suse.cz, mliska@suse.cz

Outlilne

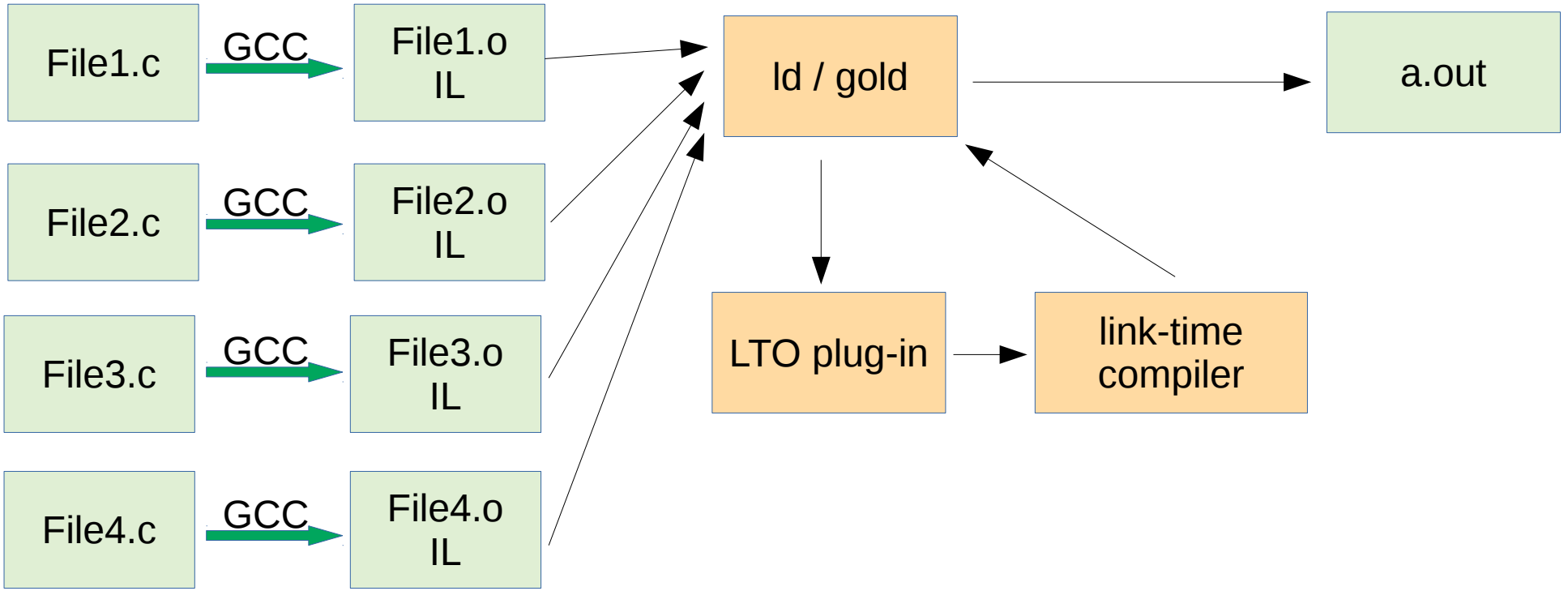
- What is link-time optimization?
- Link-time optimization and GCC
- Benchmarks
- Can we build openSUSE with link-time optimization by default?

What is link-time optimization?

Per-file compilation



Link-time optimization



Benefits of LTO

- Symbol promotion
  (from linker's resolution data most symbols become "static")
- Cross-module inlining, constant propagation
- Aggressive unreachable code removal
- Profile propagation
- EH optimization (propagation of "nothrow")
- Identical code folding
- Optimized code layout
- and more :)

Problems of LTO

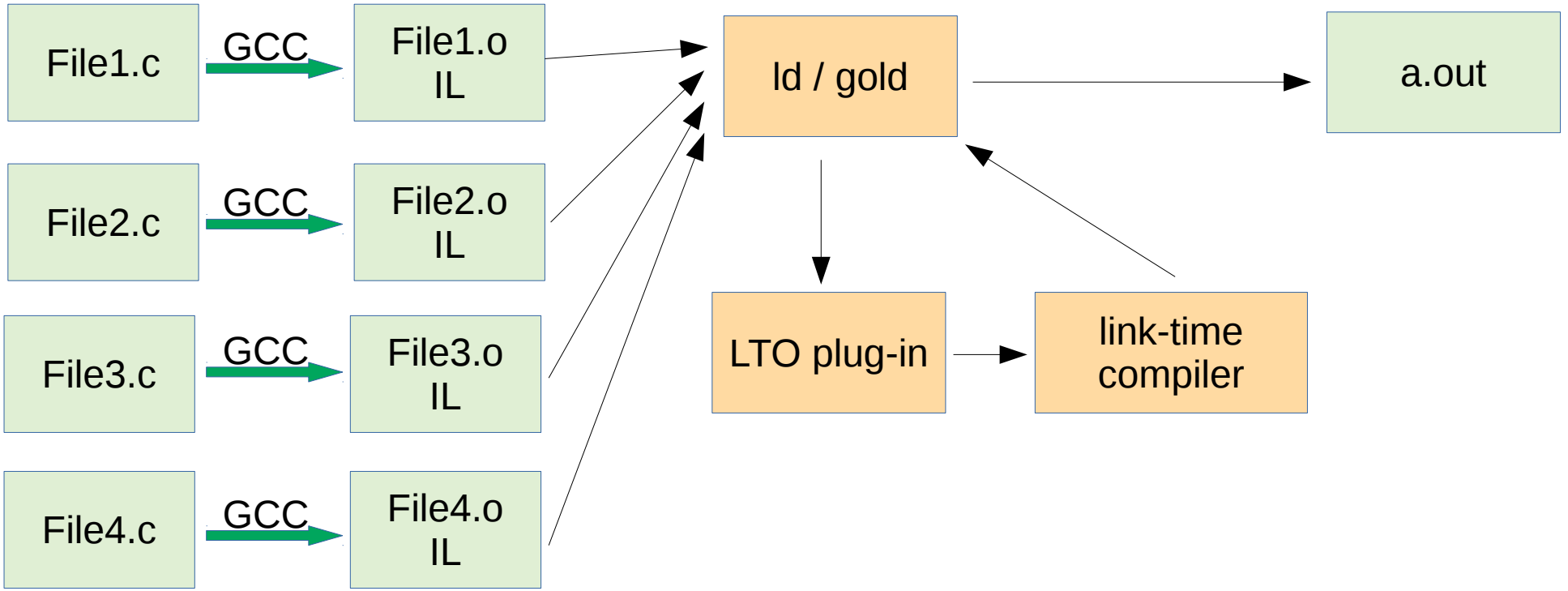
- Whole toolchain has to be restructured
- Slow compile-edit cycle
- Harder bugreports
  (often whole program needed to reproduce issue)
- Not 100% transparent to user, but in most cases all one needs to do is to add -flto

Link-time optimization and GCC

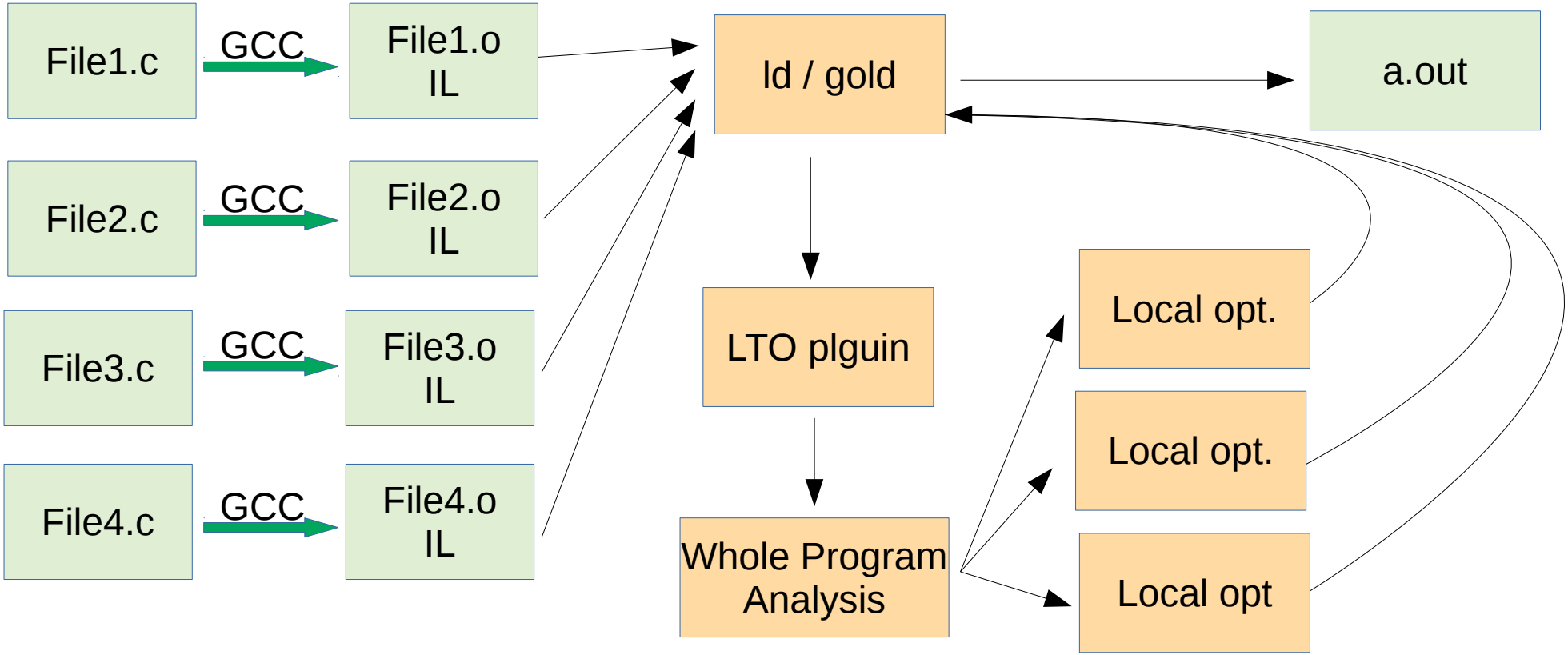
Modernizing GCC (LTO perspective)

- **1999 (GCC 2.95):** Function-at-a-time
- **2001 (GCC 3.0):** New inliner (first high-level opt . in gcc)
- **2004 (GCC 3.4):** Unit-at-a-time; intermodule compilation for C; Inter-procedural optimization framework
- **2005 (GCC 4.0):** New SSA optimization framework
- **2006 (GCC 4.1):** Inter-procedural optimizations: profile guided inlining, pure/const discovery, mod/ref, inter-procedural constant propagation, --fwhole-program –combine
- **2008 (GCC 4.4):** Inter-procedural optimization on SSA; early optimization and inlining.
- **2010 (GCC 4.5):** Basic LTO framework (5 years in development)
- **2011 (GCC 4.6):** WHOPR (parallel link-time optimization); Firefox builds

Link-time optimization



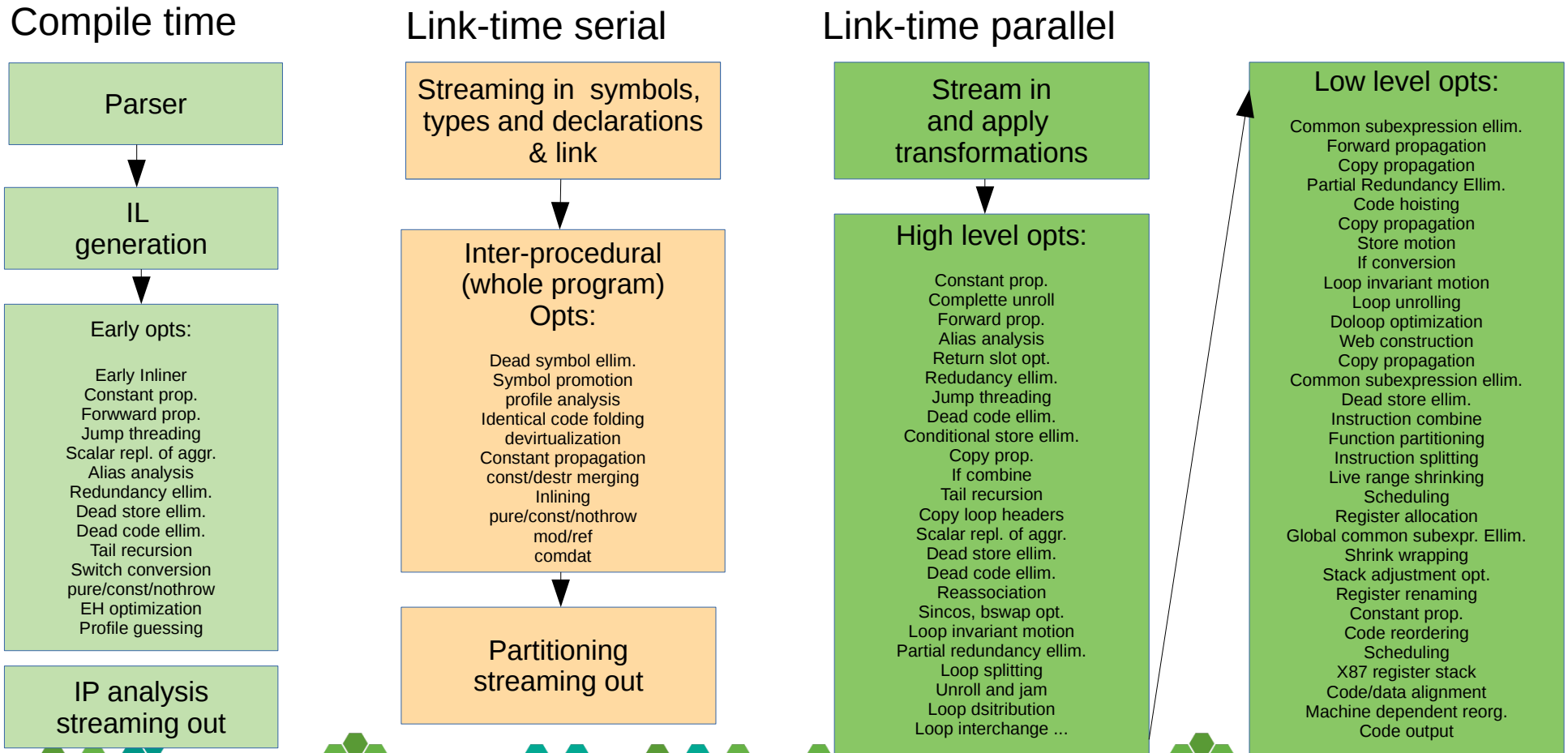
Parallelized Link-time optimization (WHOPR)



Modernizing GCC (LTO perspective)

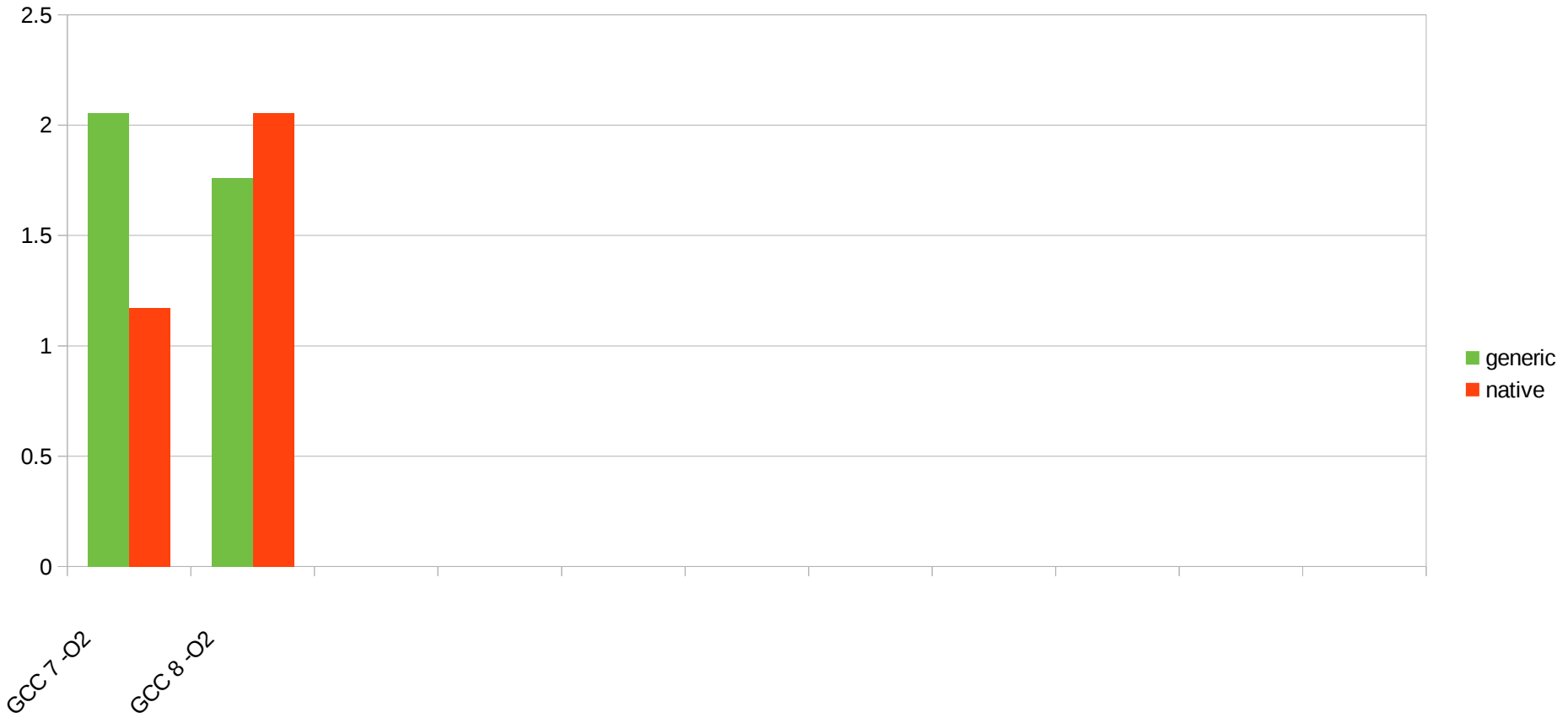
- **2012 (GCC 4.7.0):** Memory use optimizations, new inliner heuristics, new inter-procedural constant propagation with clonning
- **2013 (GCC 4.8.0):** symbol table; propagation of values passed through aggregates
- **2014 (GCC 4.9.0):** slim LTO objects by default; on demand loading of functions; devirtualization pass; feedback directed code layout
- **2015 (GCC 5):** Identical Code Folding; COMDAT optimization; One Definition Rule for C++; alignment propagation; correct command line options handling with LTO
- **2016 (GCC 6):** Linker-plugin now detects type of output binary. C&Fortran type merging. Better alias anaysis
- **2017 (GCC 7):** Inter-procedural value range propagaion; bitwise propagation
- **2018 (GCC 8):** Early debug info. Profile representation rewrite. Function splitting now by default. Reworked runtime estimation; Malloc attribute propagation

GCC optimization pipeline

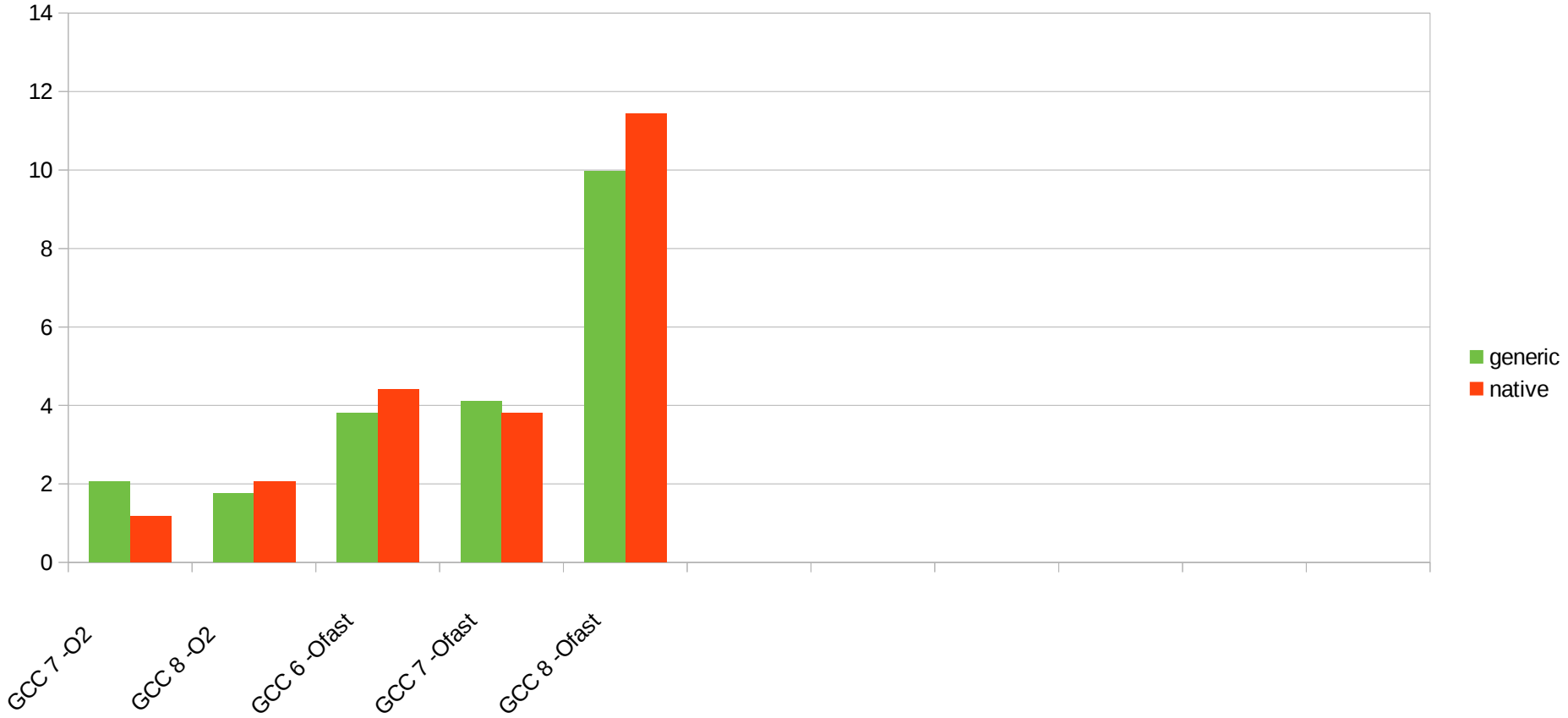


Benchmarks

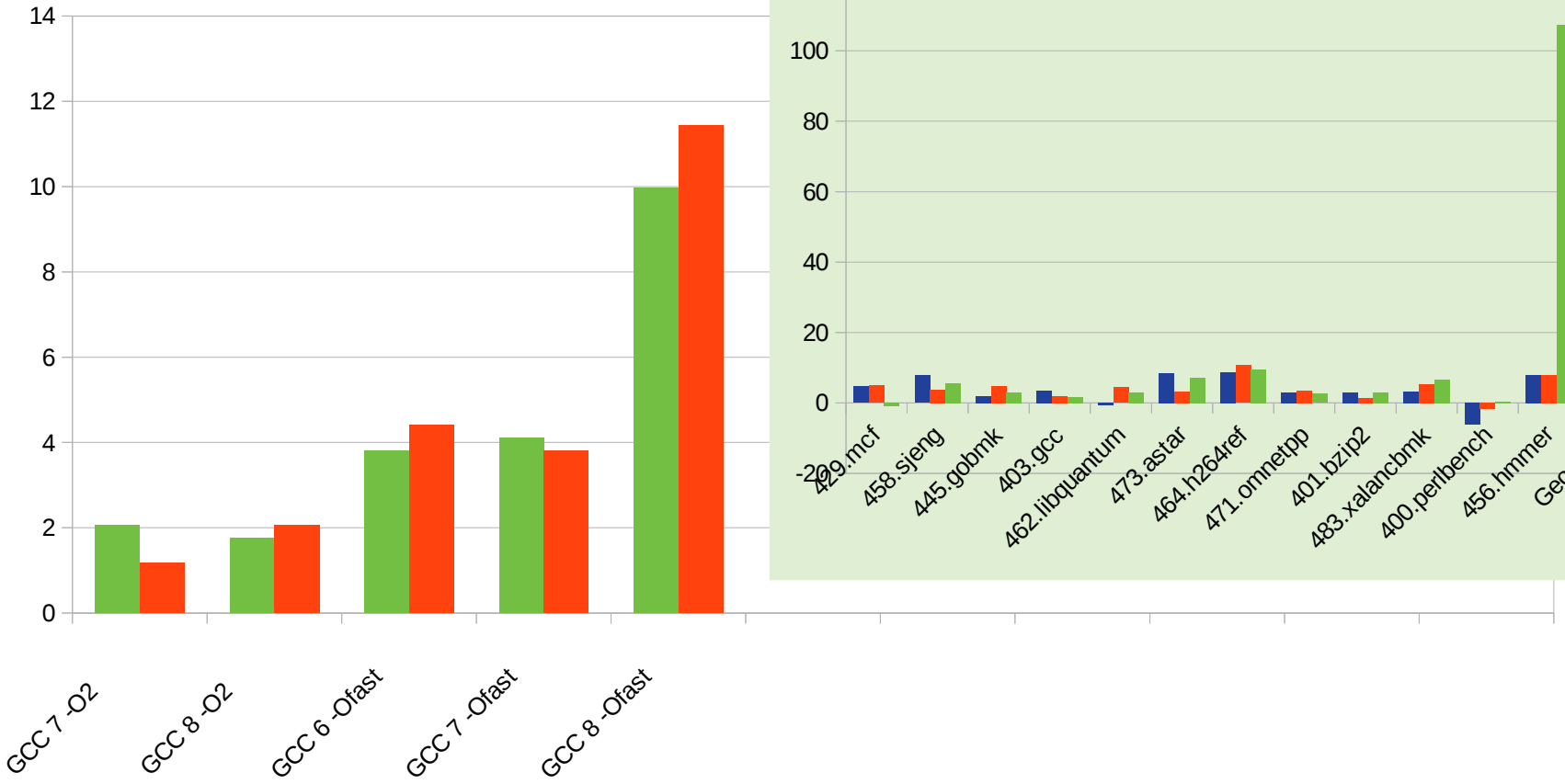
SPECint 2006 performance (relative to GCC 6)



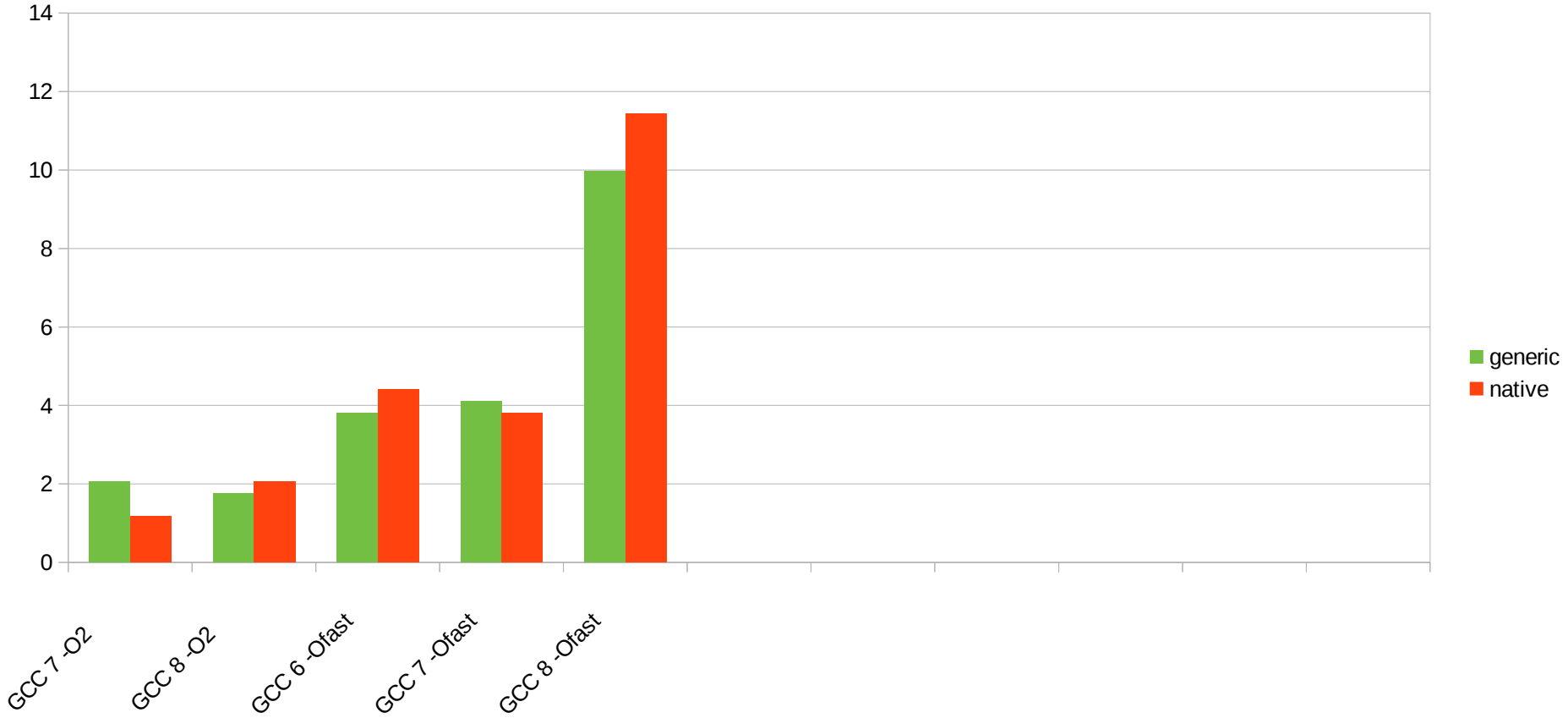
SPECint 2006 performance (relative to GCC 6)



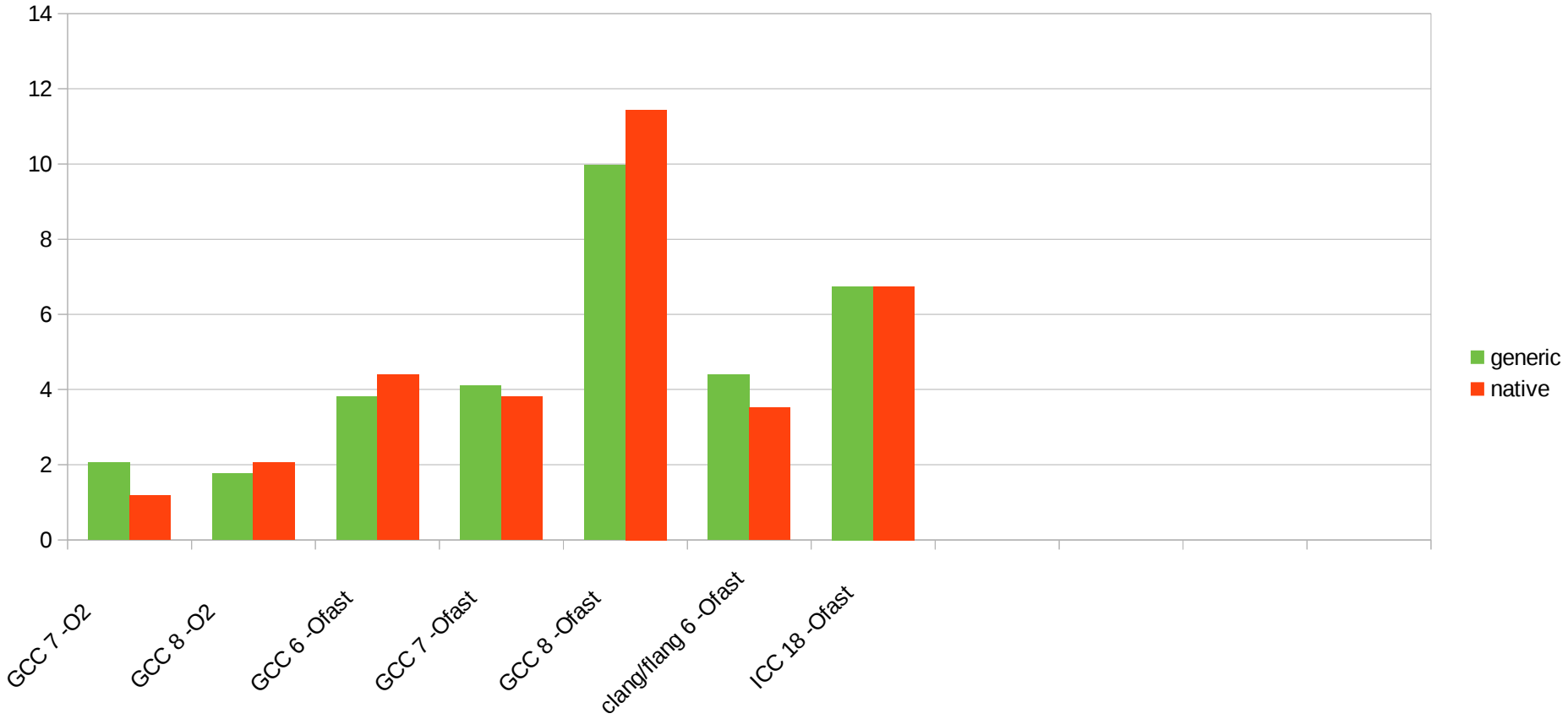
SPECint 2006 performa



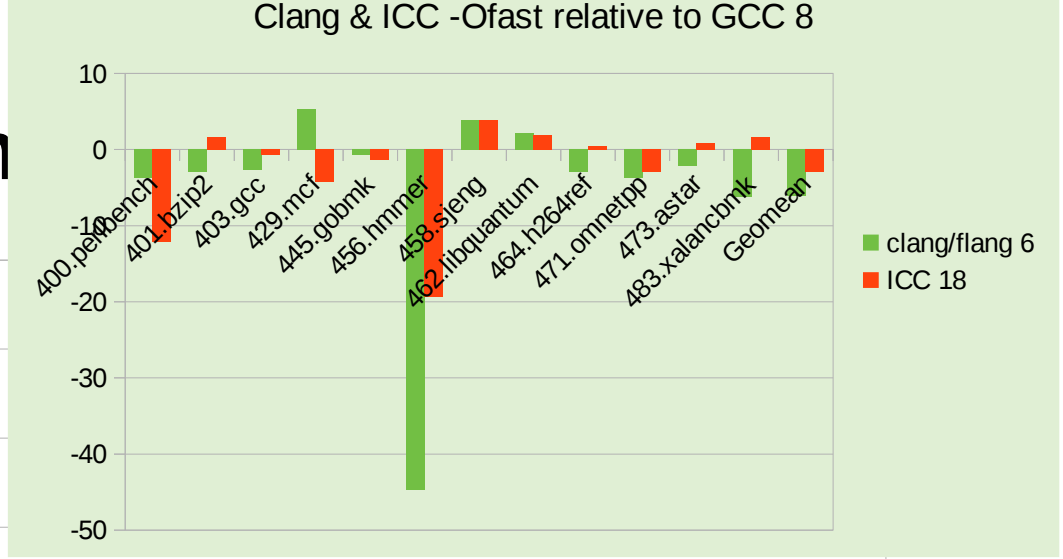
SPECint 2006 performance (relative to GCC 6)



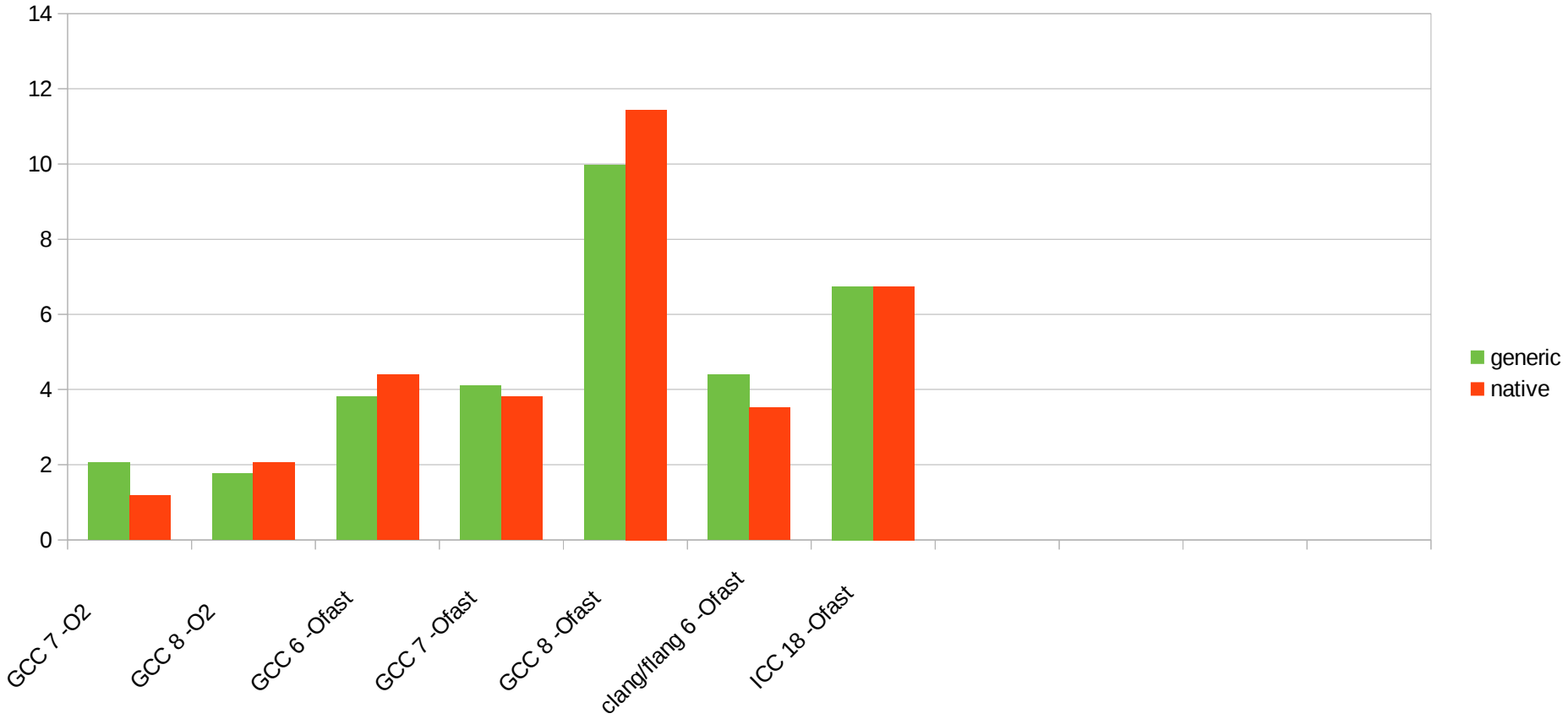
SPECint 2006 performance (relative to GCC 6)



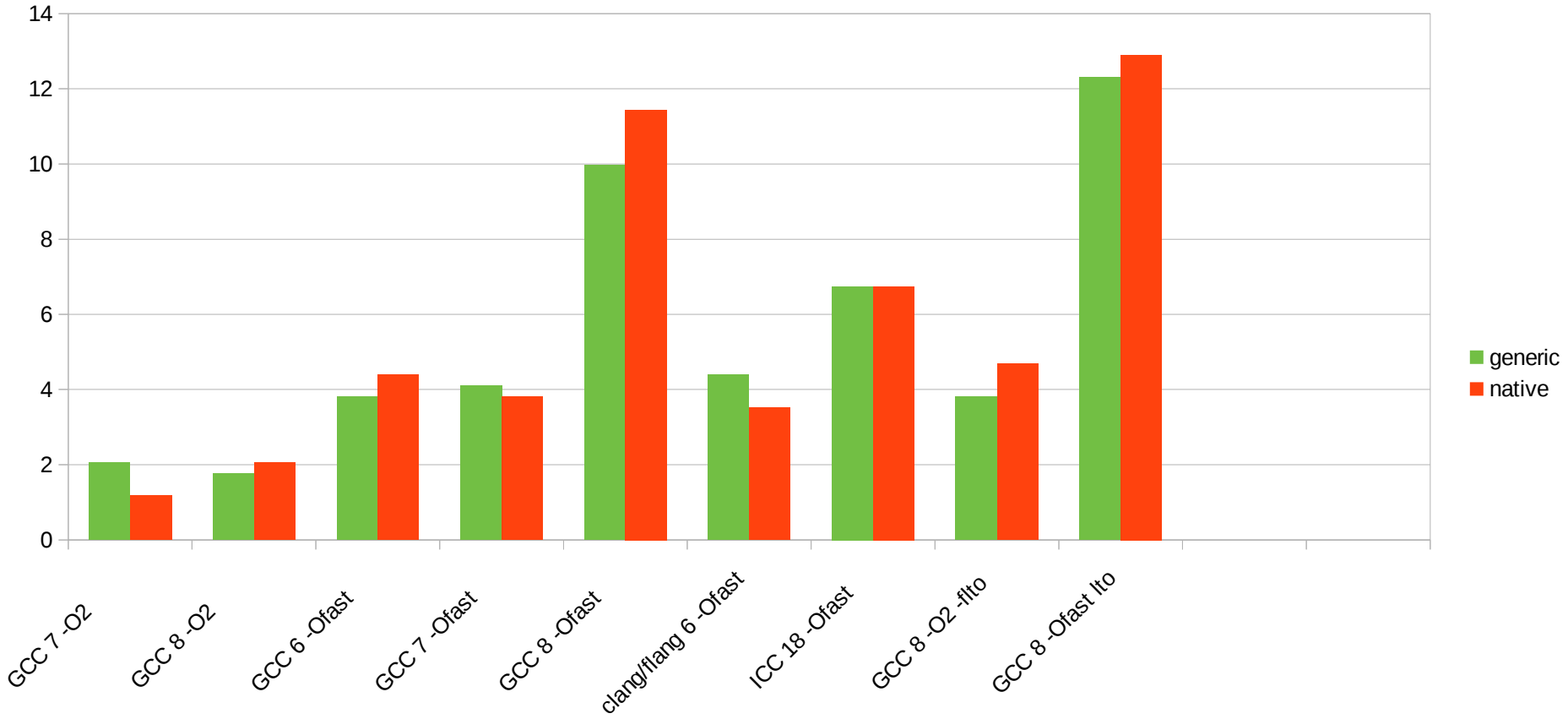
SPECint 2006 perform



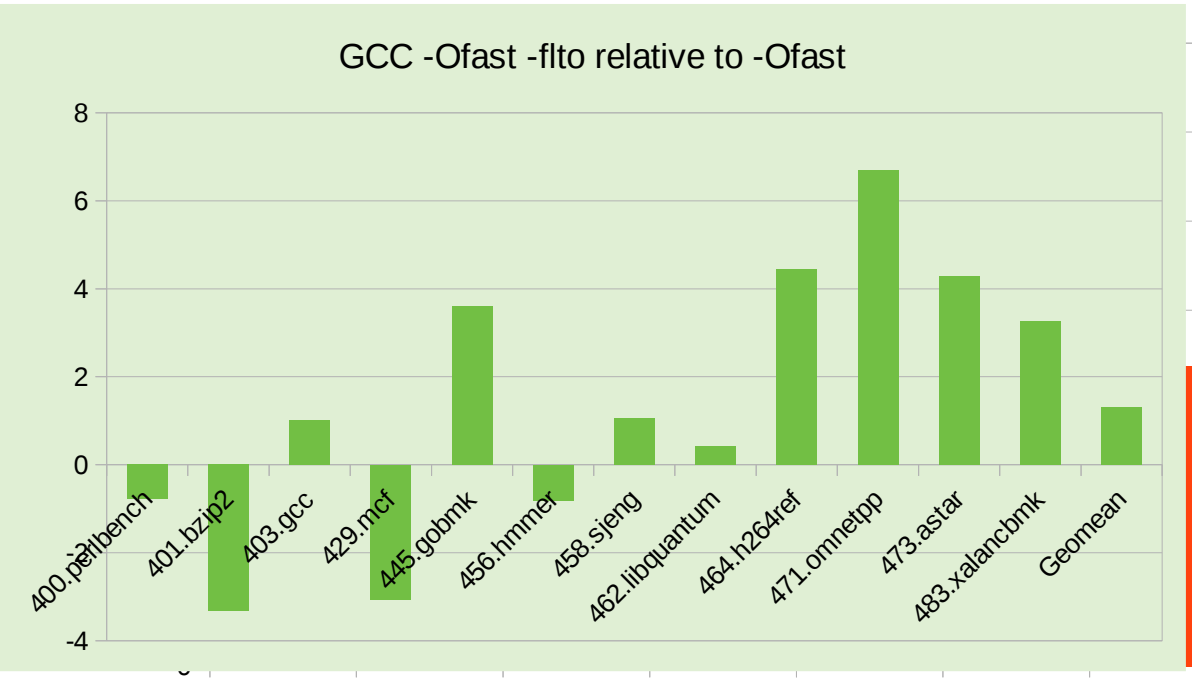
SPECint 2006 performance (relative to GCC 6)

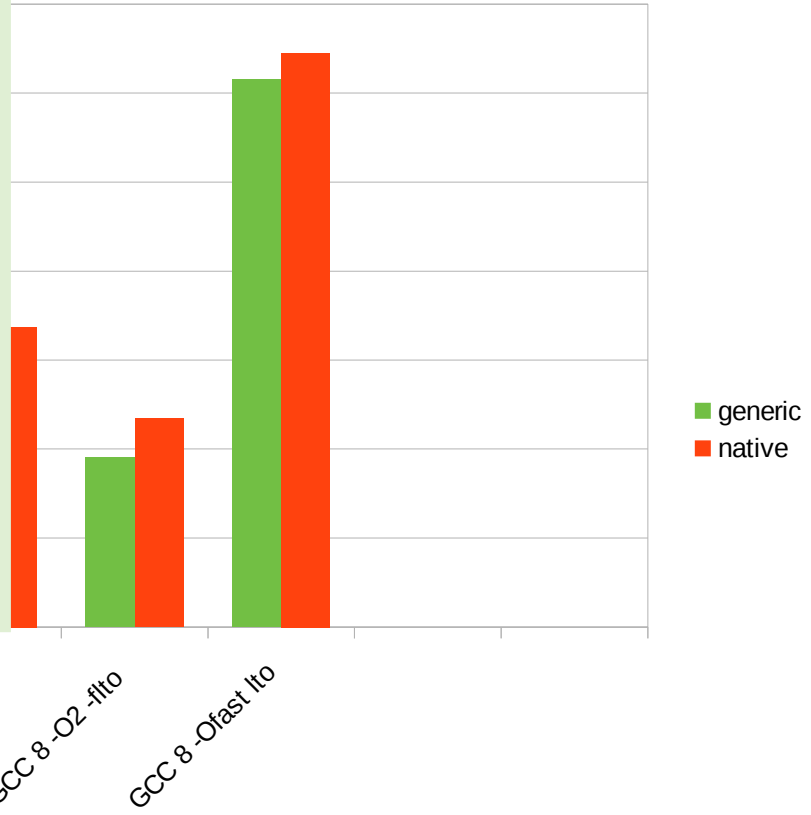


SPECint 2006 performance (relative to GCC 6)

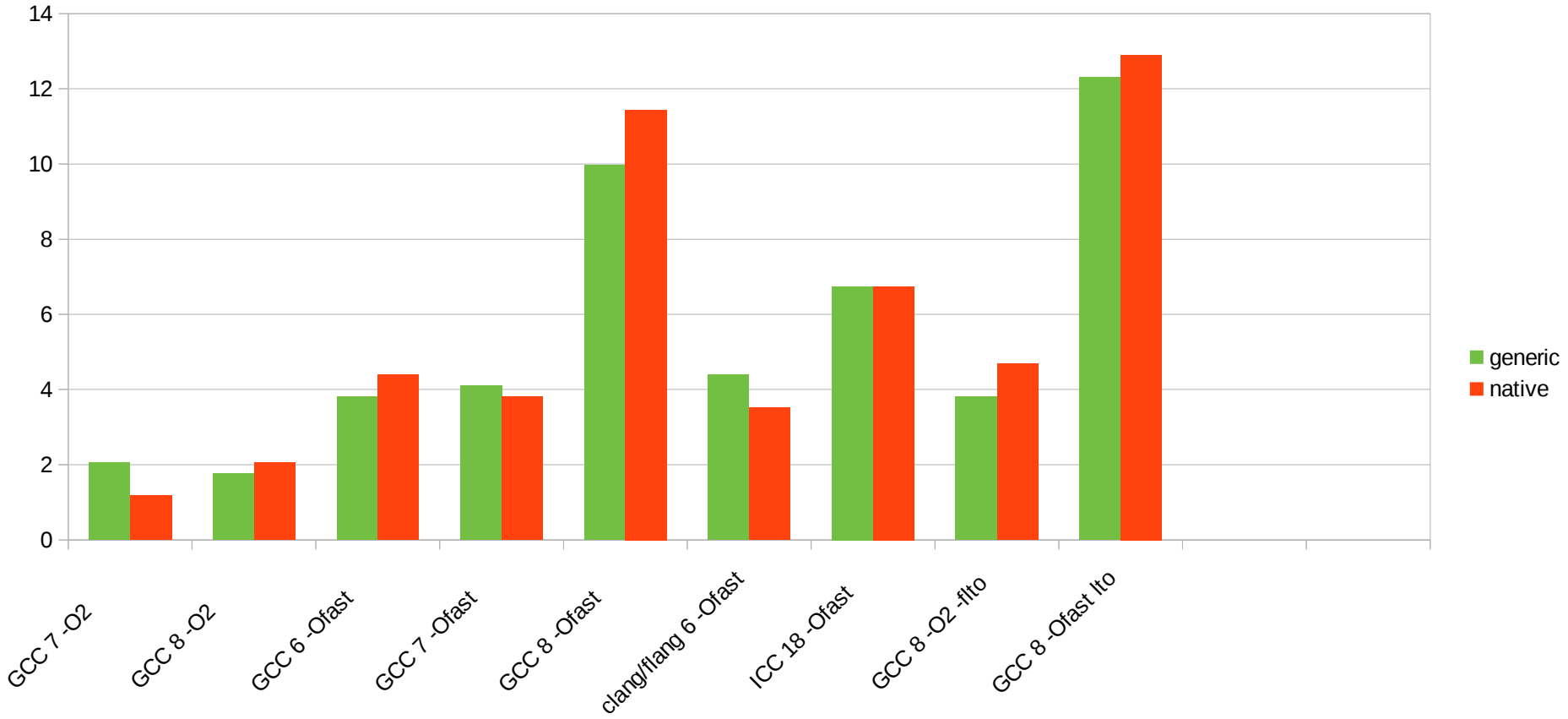


SPECint 2006 performance (relative to GCC 6)

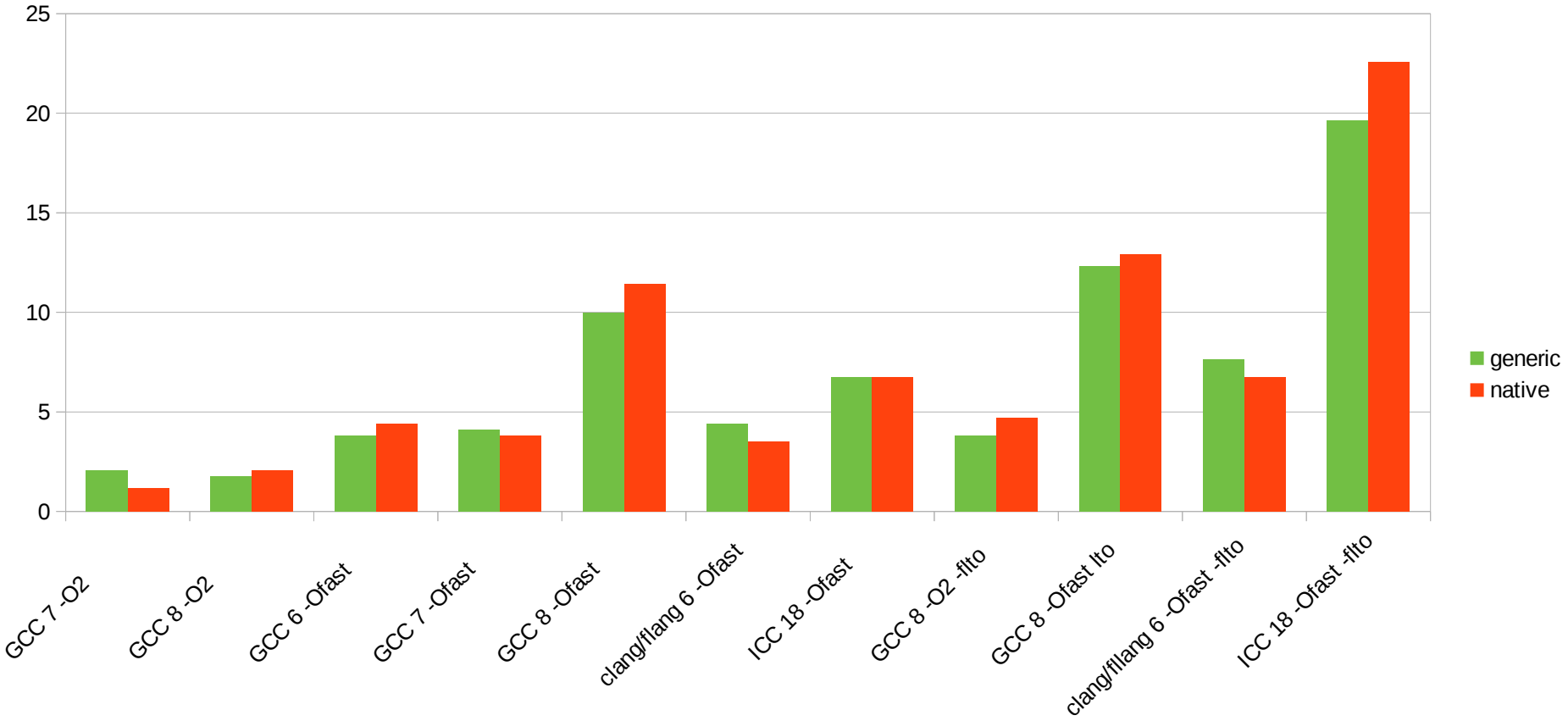




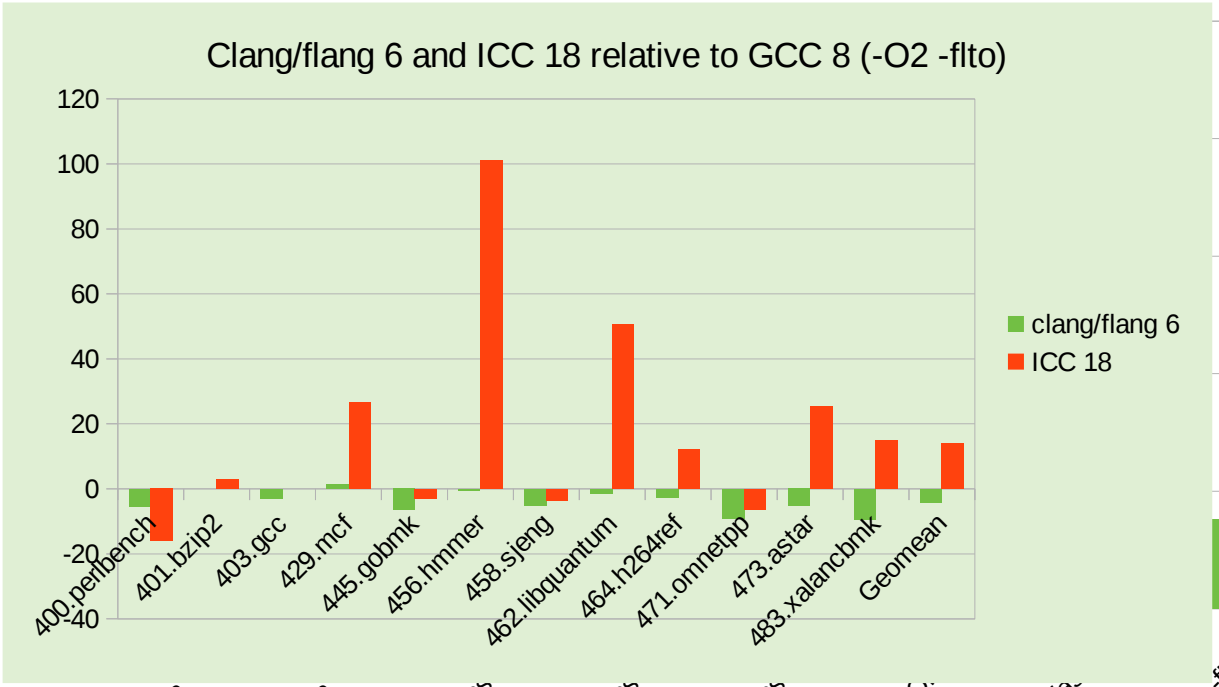
SPECint 2006 performance (relative to GCC 6)

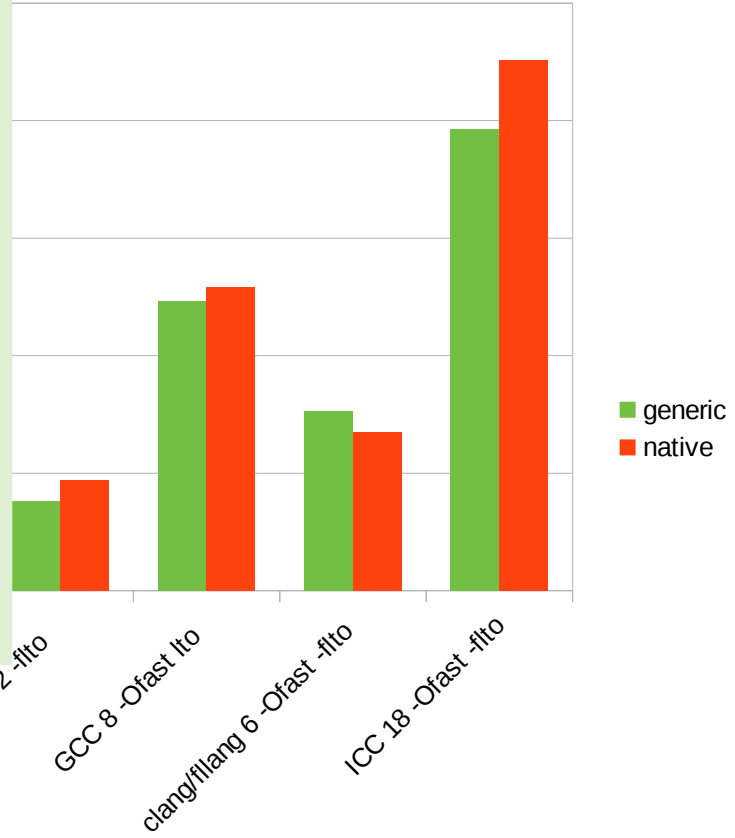


SPECint 2006 performance (relative to GCC 6)



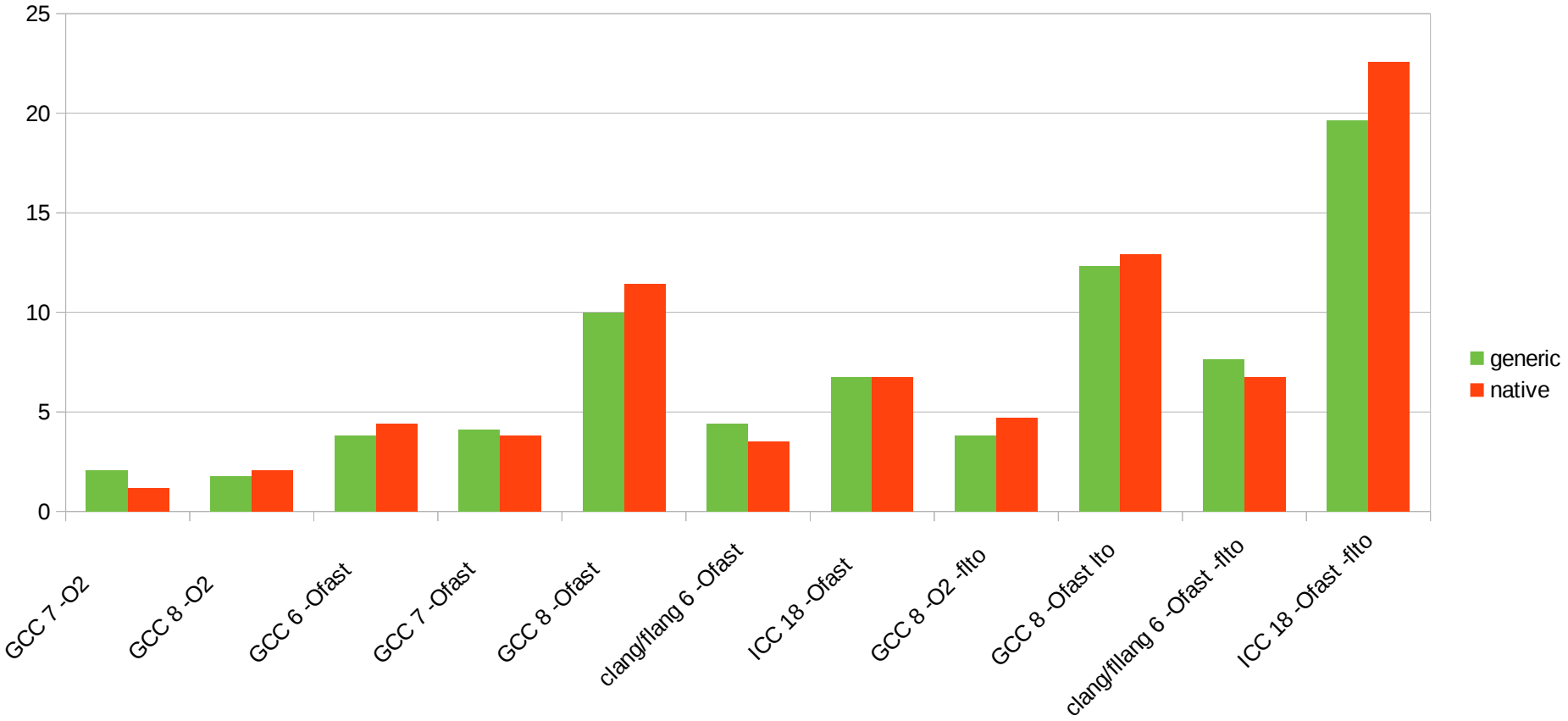
SPECint 2006 performance (relative to GCC 6)



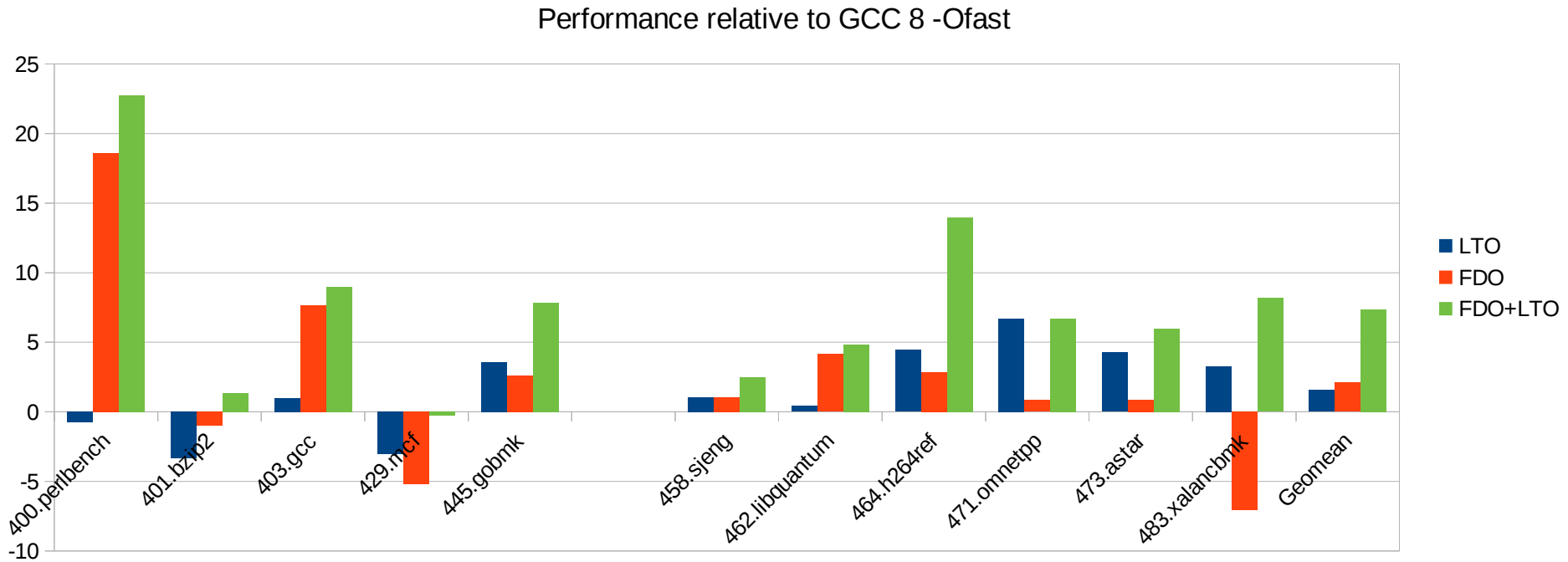


GCC 7 -O2
GCC 8 -O2
GCC 6 -Ofas
GCC 7 -Ofas
GCC 8 -Ofas
clang/flang 6 -O
ICC 18 -Ofa

SPECint 2006 performance (relative to GCC 6)



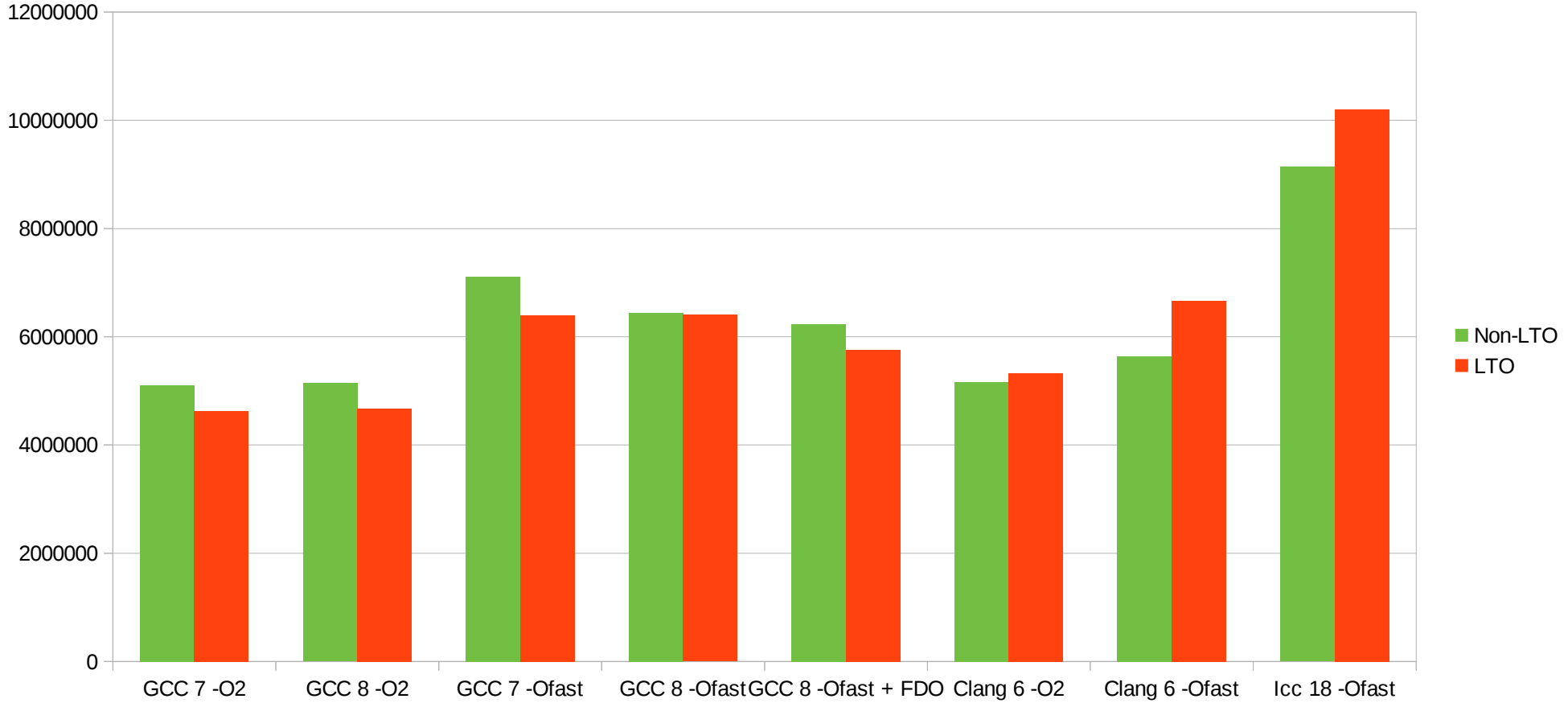
Profile feedback & LTO works well together



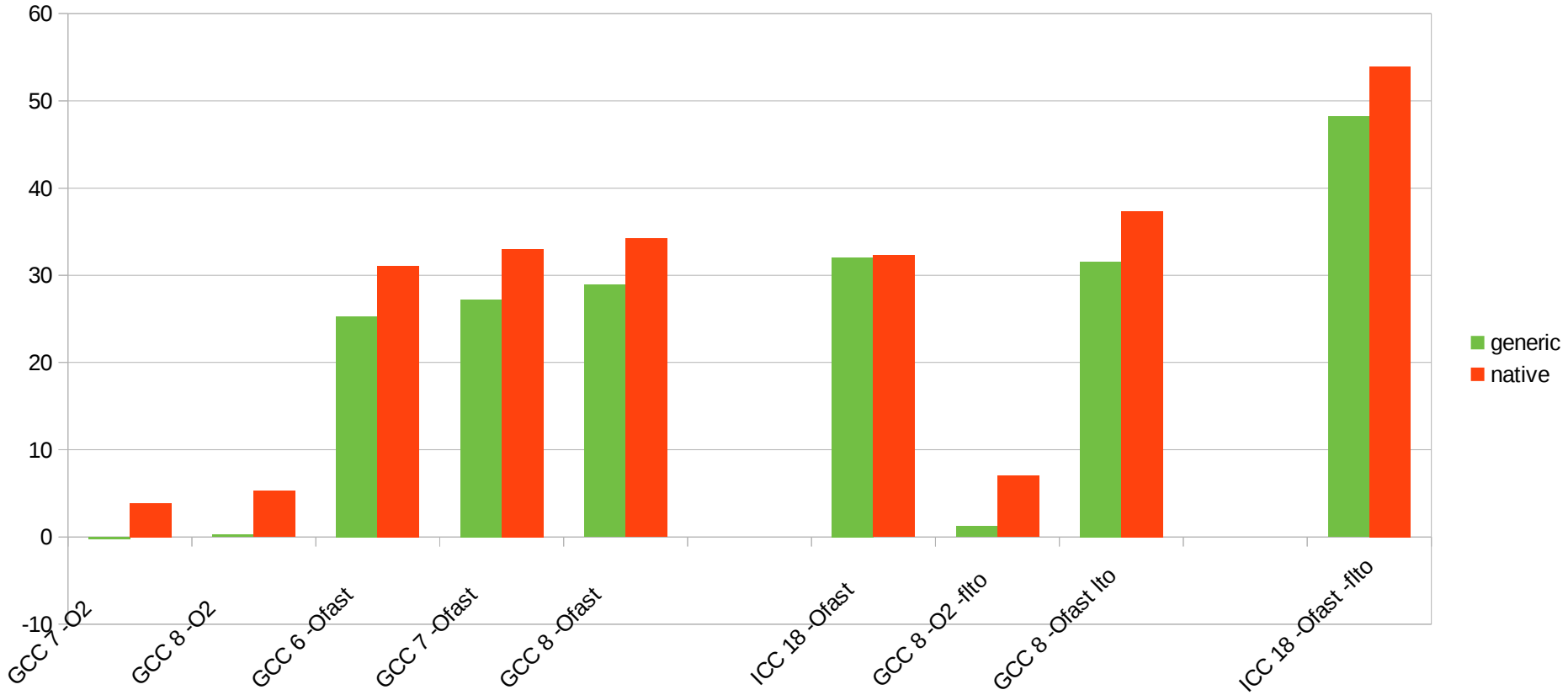
Hmmer benchmark has problems; was excluded.

To build with profile feedback often you can use:
./configure ; CFLAGS=”-O2 -fprofile-generate” make ; make check ; make clean ; CFLAGS=”-O2 -fprofile-use” make

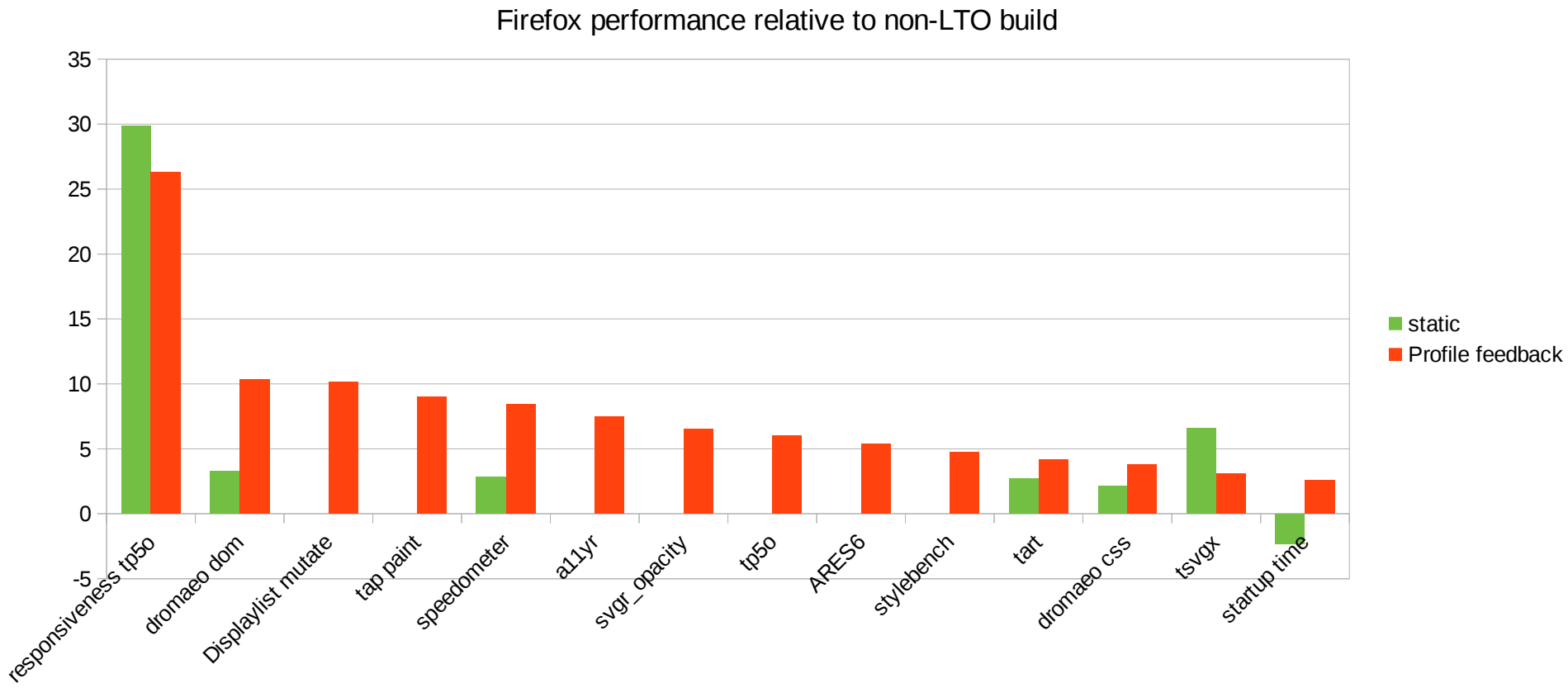
SPECint 2006 code size



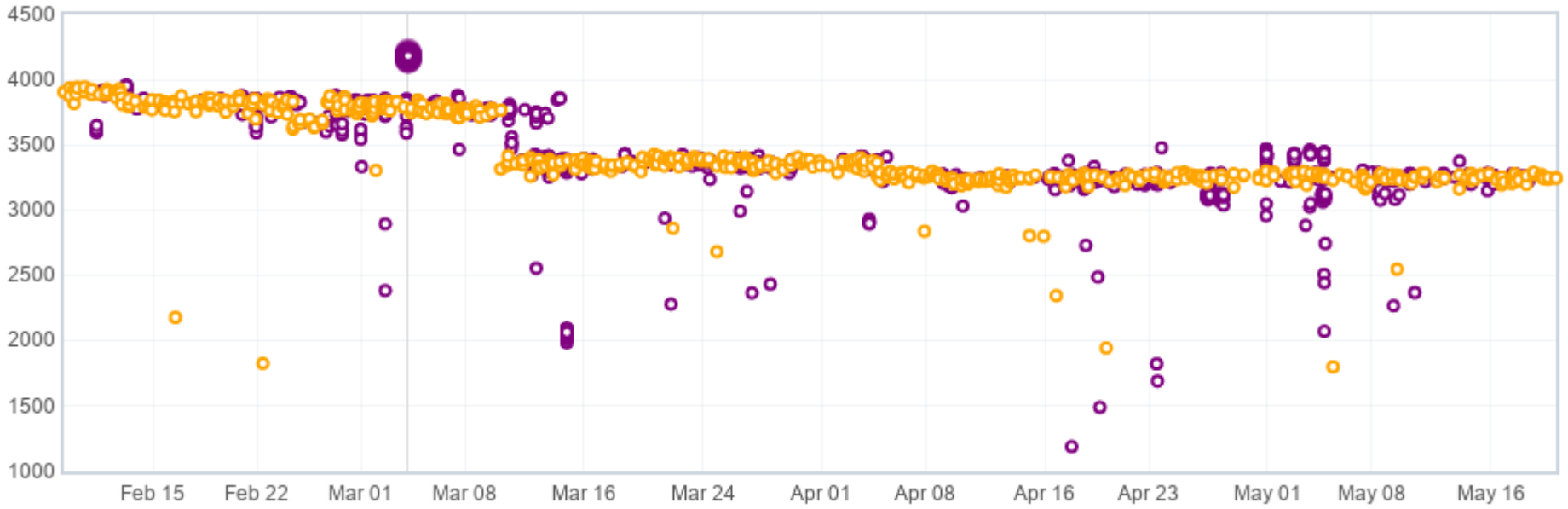
SPECfp 2006 performance (relative to GCC 6)



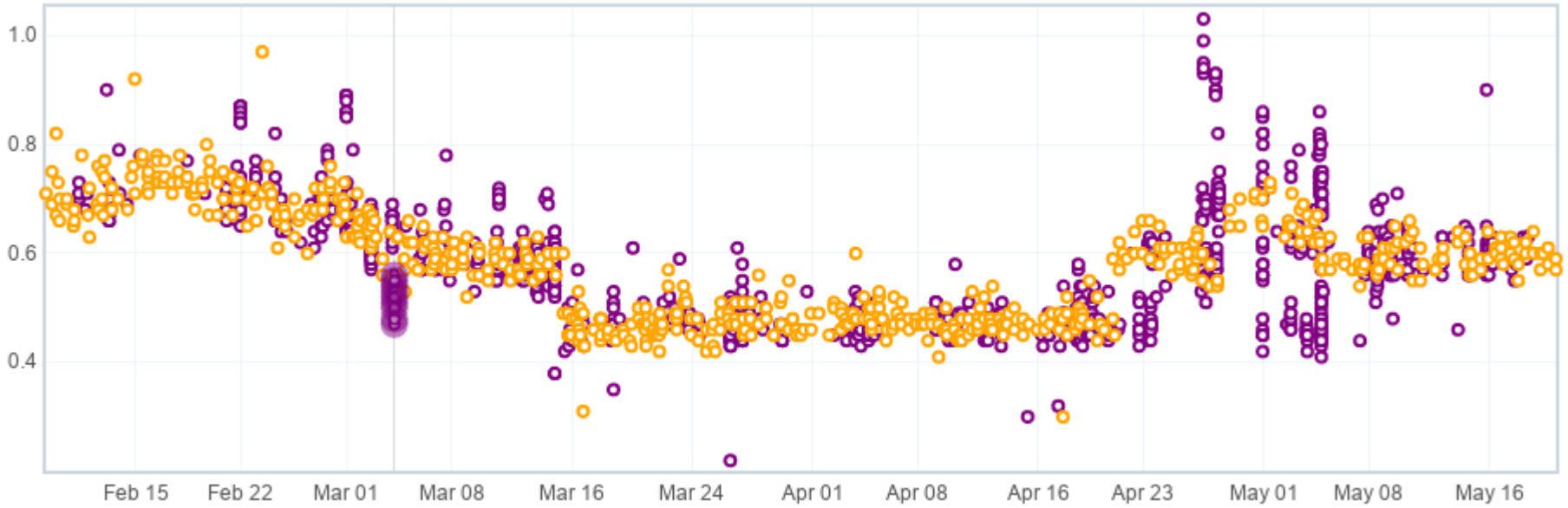
Firefox performance summary



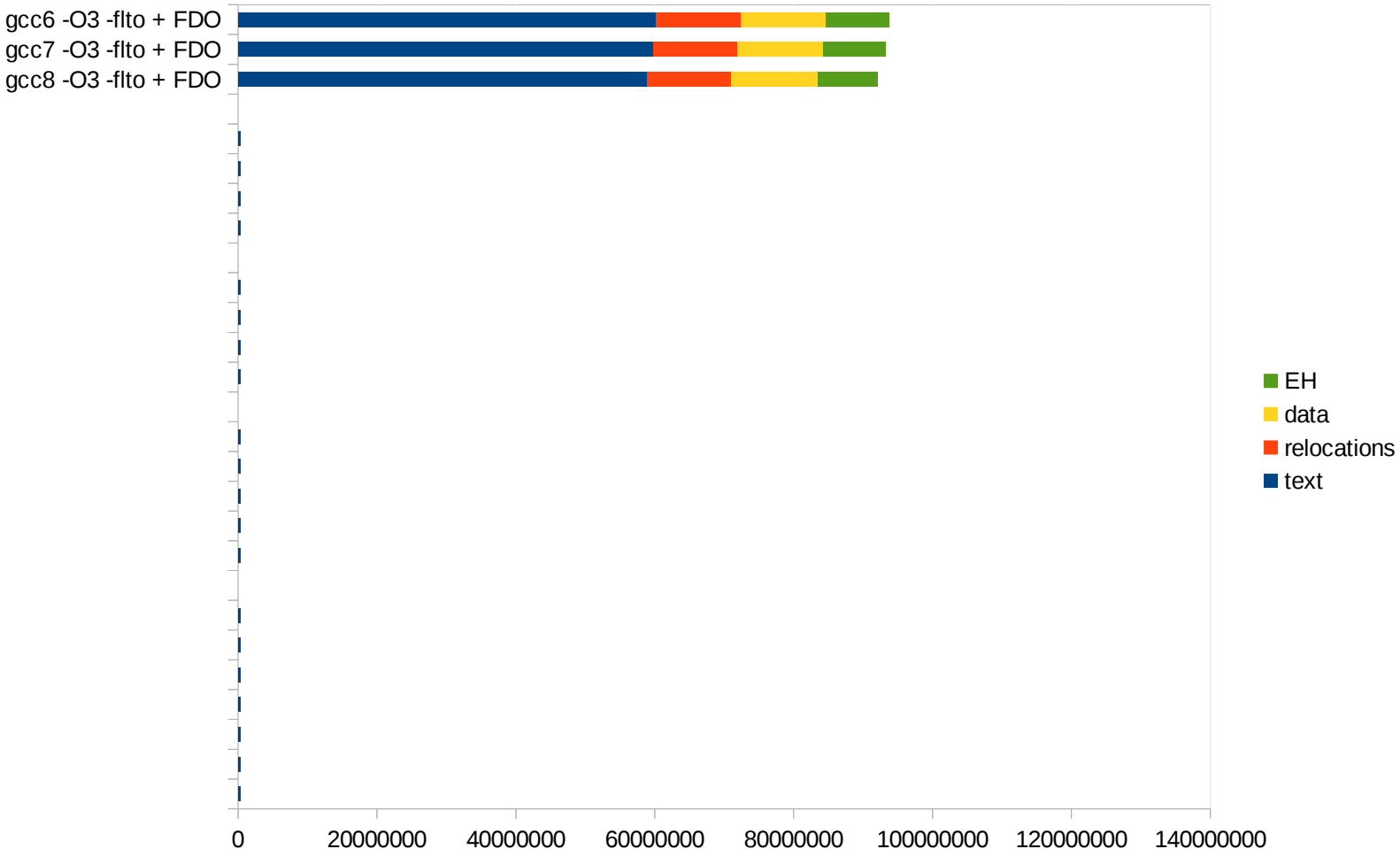
Firefox performance – dromaeo DOM



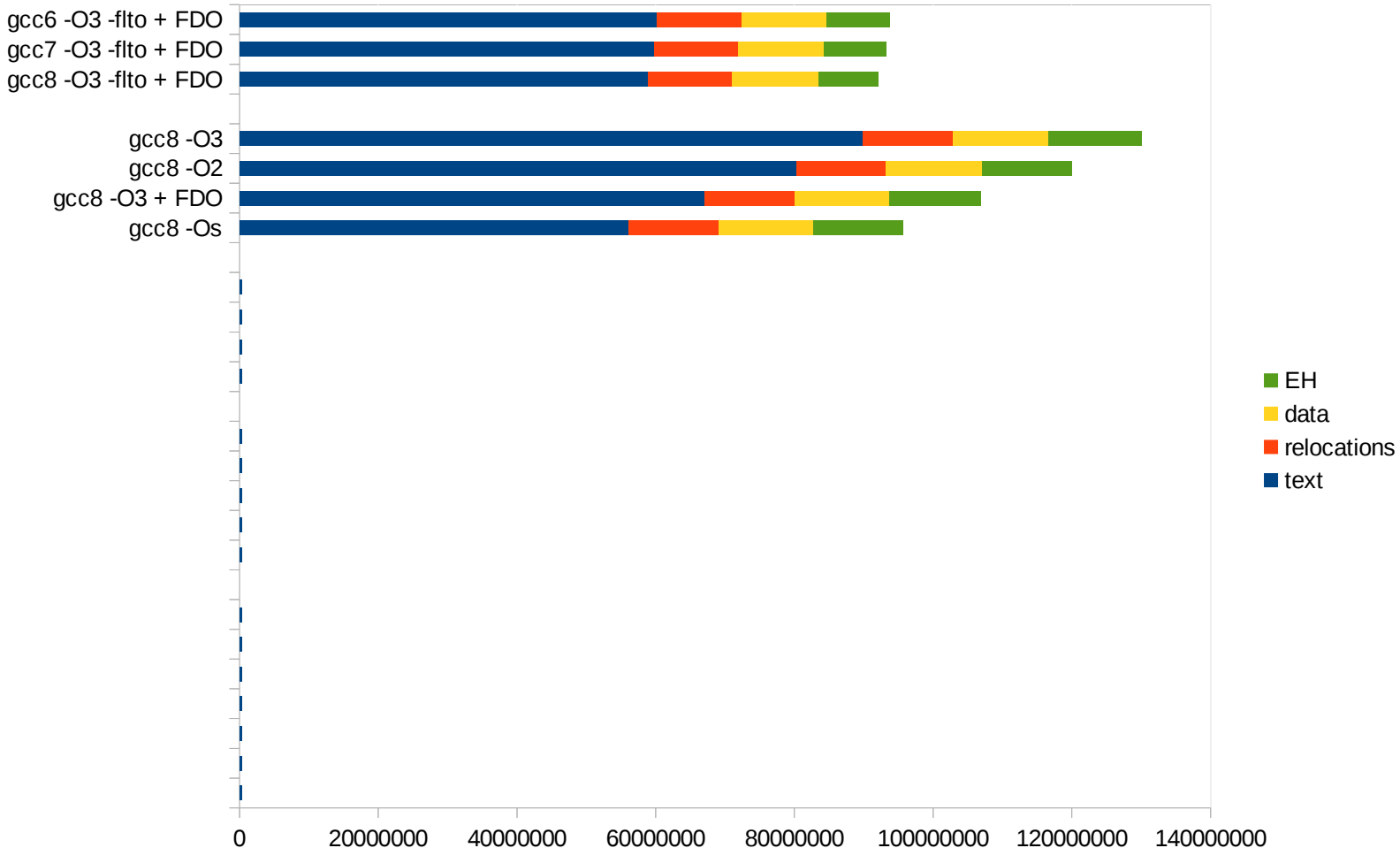
Firefox performance – tp5o page responsiveness



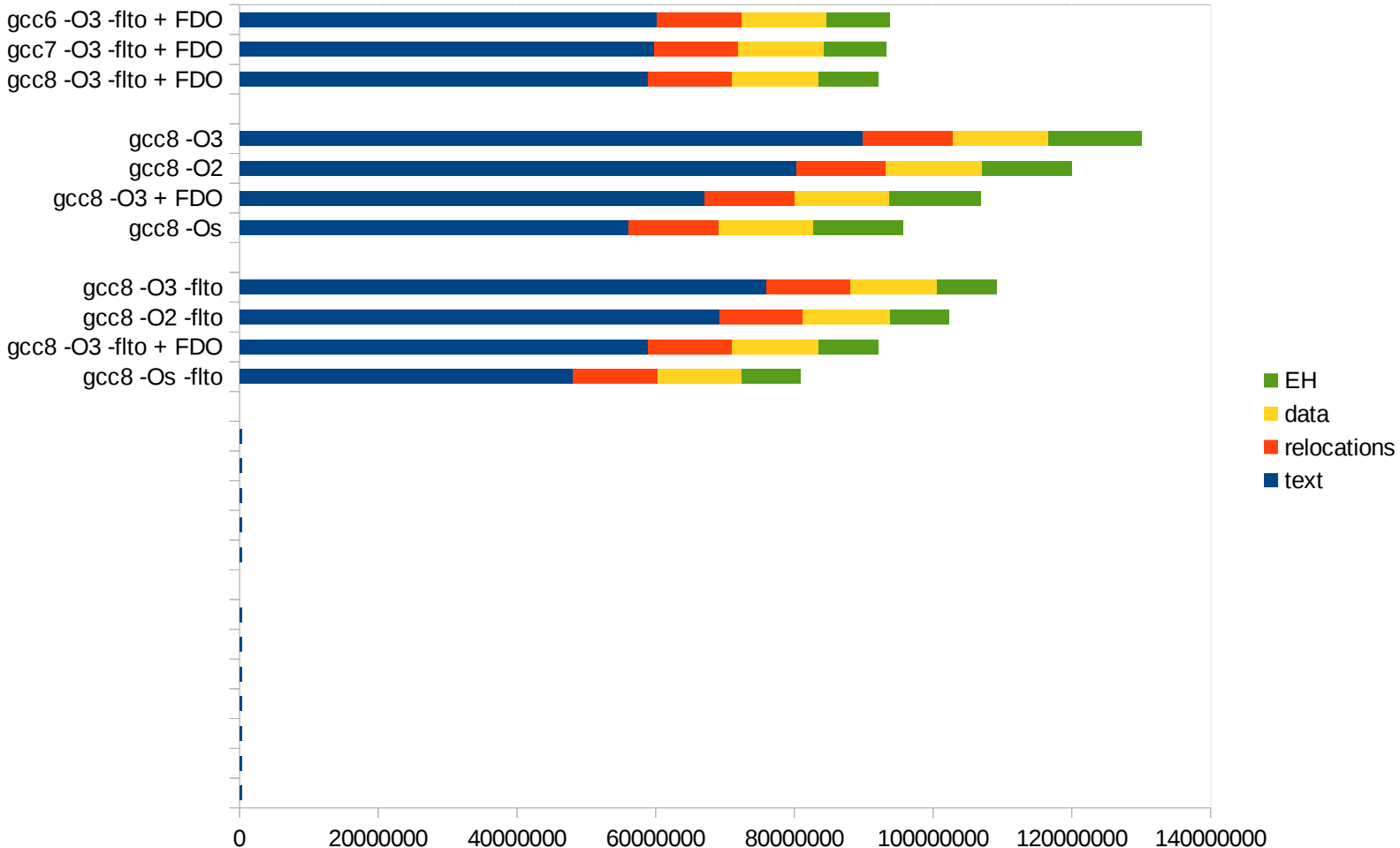
Firefox binary size



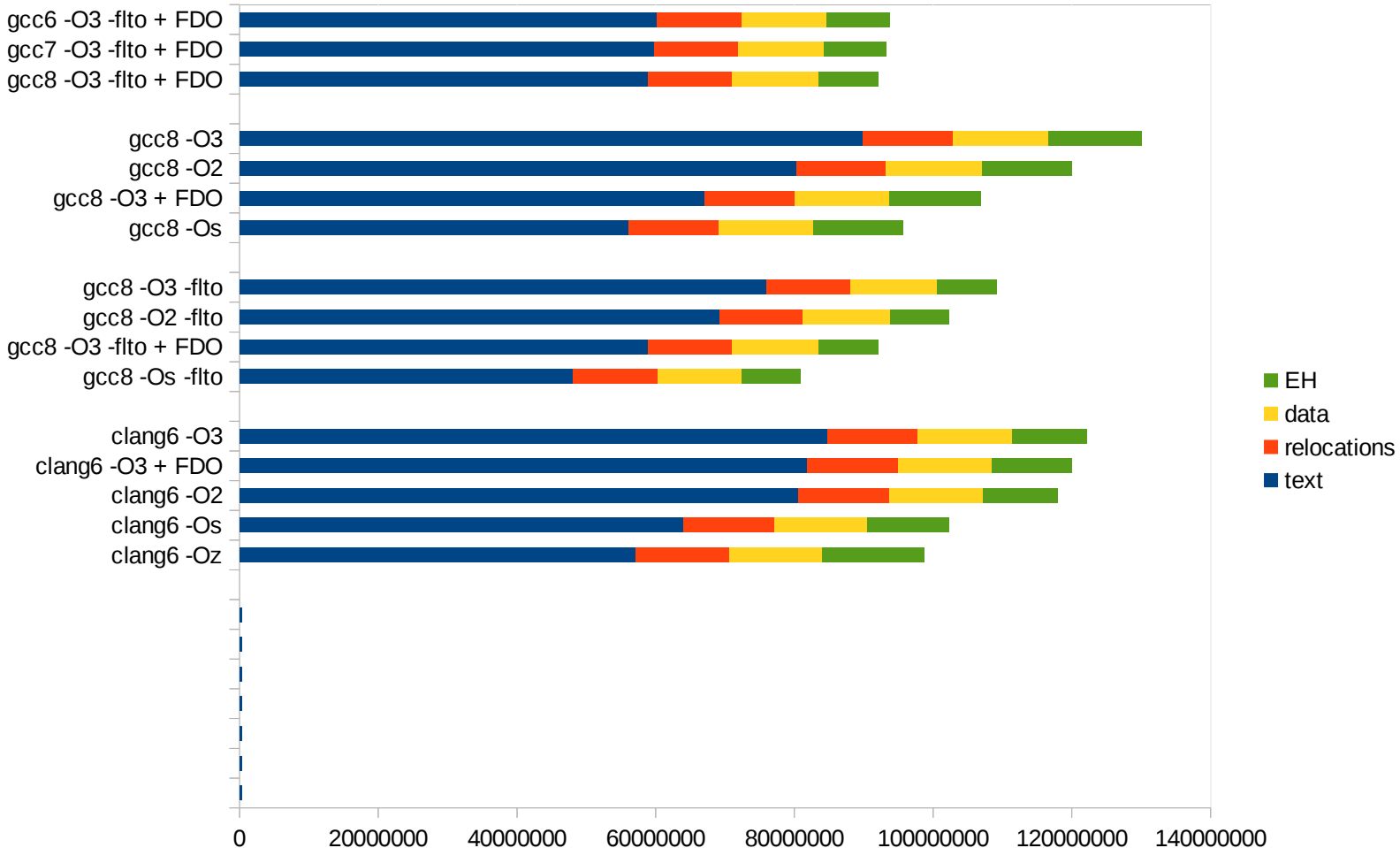
Firefox binary size



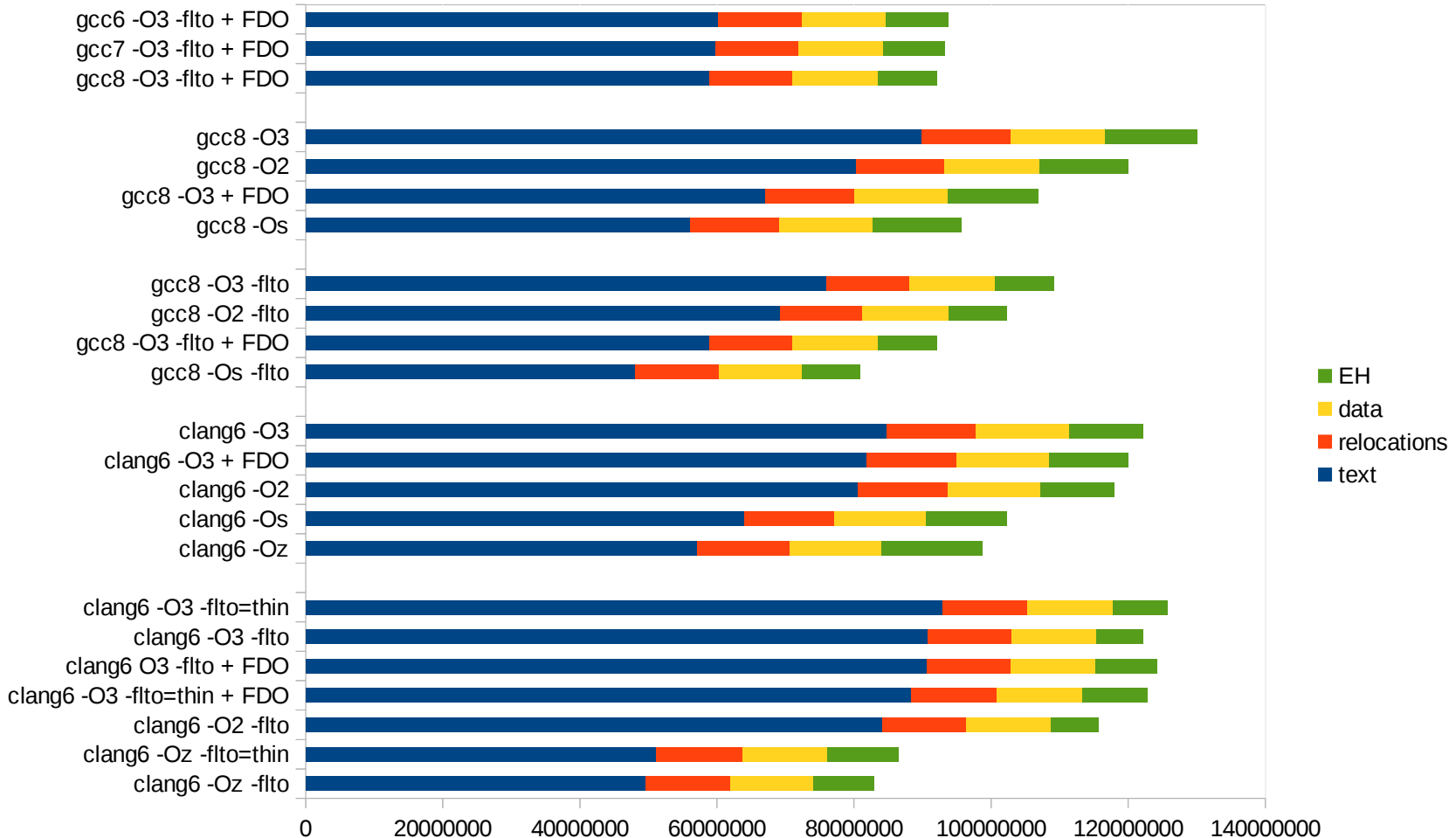
Firefox binary size



Firefox binary size



Firefox binary size



Summary

- LTO now works and scale for large applications
  (Firefox, Libreoffice,…)
- Important size optimization especially for large C++ programs
- Often important performance optimization especially when combined with profile feedback
- Space for future improvements
  (both on GCC side and re-optimization of programs to LTO model)

LTO in openSUSE Factory

LTO in openSUSE Factory

- A new staging project (openSUSE:Factory:Staging:N)
- Setting: *Optflags: * -flto*
- ~80 packages failed (out of all ~2300)
- Branch all failing packages and disable LTO
- LTO Factory (w/ some disabled packages) can run openQA test-suite (with a small fallout)

Package statistics

- Total ELF files in distribution: ~6700
- Reduction: 1839 MB to 1736MB (-5.6%)
- libmergedlo.so: reduction 10MB (-16%)
- Biggest reduction:
  - mysql_upgrade (-92%)
  - Innochecksum (-94%)
  - mbstream (-94%)

Issues

- Argyllcms: lto1: fatal error: multiple prevailing defs for 'xcal_read_icc': LD PR: https://sourceware.org/PR23079
- GCC LTO miscompilation: http://gcc.gnu.org/PR85248
- *.symver* in shared libraries: https://gcc.gnu.org/PR48200:
  - __asm__(".symver old_foo,foo@VERS_1.1")
  - New *symver* function attribute must be added (GCC 9.1.0)

Issues (cont.)

- static libraries (*.la):
  - e2fsprogs, btrfsprogs, …
  - Fat LTO objects must be used (-ffat-lto-objects)
  - LTO elf sections should be stripped
  - OBS sanitizer should be extended
  - LTO mode can combine both LTO objects and assembly objects

Issues (cont. 2)

- LTO warnings (-Wodr, ...):
  - *ltrace*: error: type of 'filter_matches_symbol' does not match original declaration [-Werror=lto-type-mismatch]
  - *gdb*: error: type 'struct ipa_sym_addresses' violates the C++ One Definition Rule [-Werror=odr]
- mpir: weak configure script that scans *.o file as a blob
- weak support for LTO debug info in *dwz* tool
- Maybe higher memory constaints for selected packages

Issues (cont. 3)

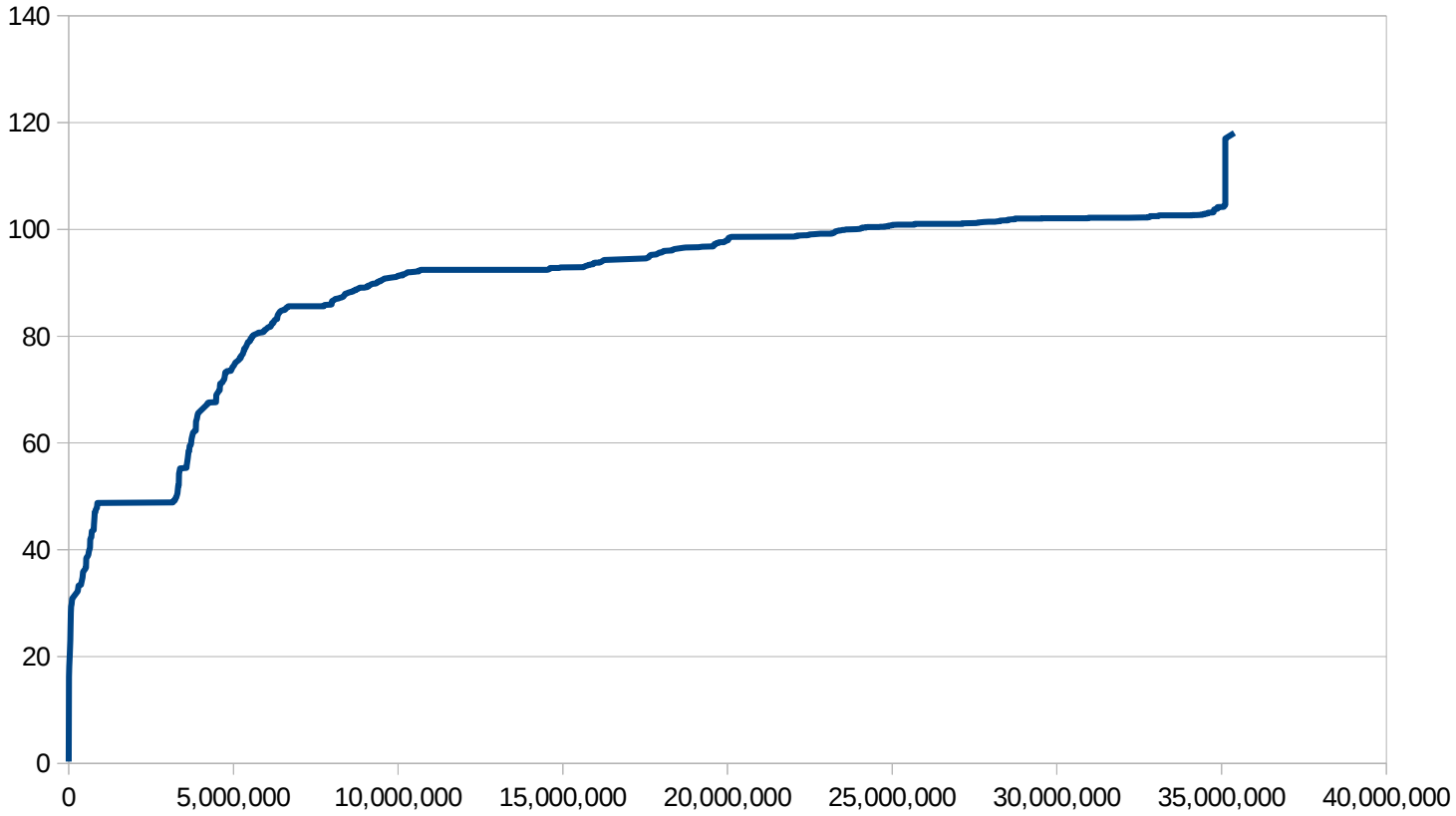
- Usage of top-level assembler (https://gcc.gnu.org/PR57703):
  - Example of *syscall.cc* in Chromium project:

```
asm volatile(".text\n"
    ".align 16, 0x90\n"
    ".type SyscallAsm, @function\n"
  "SyscallAsm:.cfi_startproc\n"
```

  - Error:*nacl_helper.ltrans1.ltrans.o: In function `playground2::SandboxSyscall(int, long, long, long, long, long, long)':*

    nacl_helper.ltrans1.o:(.text+0x4503): undefined reference to `SyscallAsm'

Histogram of text segment sizes



Conclusion

- LTO looks mature enough to be used by default
- Do we want it for openSUSE Factory?
- For the future, can we offer it also for SLE?

License

General Disclaimer

Credits

Template
Richard Brown
rbrown@opensuse.org

Design & Inspiration
openSUSE Design Team
http://opensuse.github.io/branding-guidelines/