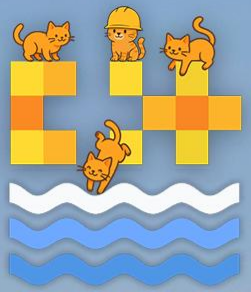


Mind the Gap

(Between Your Code and Your Toolchain)

Yannic Staudt

2025


tipi.build
by EngFlow

What’s in here?

THE DEBUG SYMBOL'S GUIDE TO THE GALAXY

Mostly harmless, occasionally symbol-less.

tipi.build by EngFlow

DWARF, the Linker, and Everything

- You start to have problems with linking typically when your binary reaches ~2GB
- By default, **gcc** and **clang** assume `-mcmodel=small`

> "*the program and its symbols must be linked in the lower 2 GB of the address space*"

Exceed that and you get all sorts of *funny* errors

Exacerbated by debug symbols, lesser levels build optimizations, etc.…

Don't Panic – Enter Debug Fission

Can be done at compile time – just add `-gsplit-dwarf` to your compiler invocation

DONE!

Even easier handling when you build yourself a debug symbol package (“DWARF package or .dwp) that contains all the symbols from your programs as you ship it.

If you want a great read-up on that topic:
https://maskray.me/blog/2022-10-30-distribution-of-debug-information

DON’T PANIC

Enabling Debug Fission

GCC:

```
g++ -g -gsplit-dwarf myfile.cpp
```

Clang:

```
clang++ -g -gsplit-dwarf myfile.cpp
```

Optional:

```
dwp myfile.o -o combined.dwp
```

Remember

The answer to *“Why is my linker screaming?”* is rarely 42.

It’s probably just too many symbols.

Use debug fission. **Don’t Panic.**

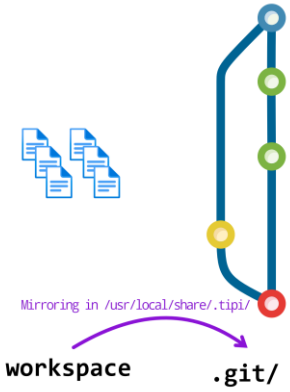
THE GREAT BRITISH BUILD-OFF

The judges are looking for a light, flaky cache with minimal linking.

tipi.build by EngFlow

Caching is

tricky



Make it invariant

Less variance == higher hit rate & less work

Controlled environments

Containers are your cache's friend

Two levels of caching for maximum effectiveness

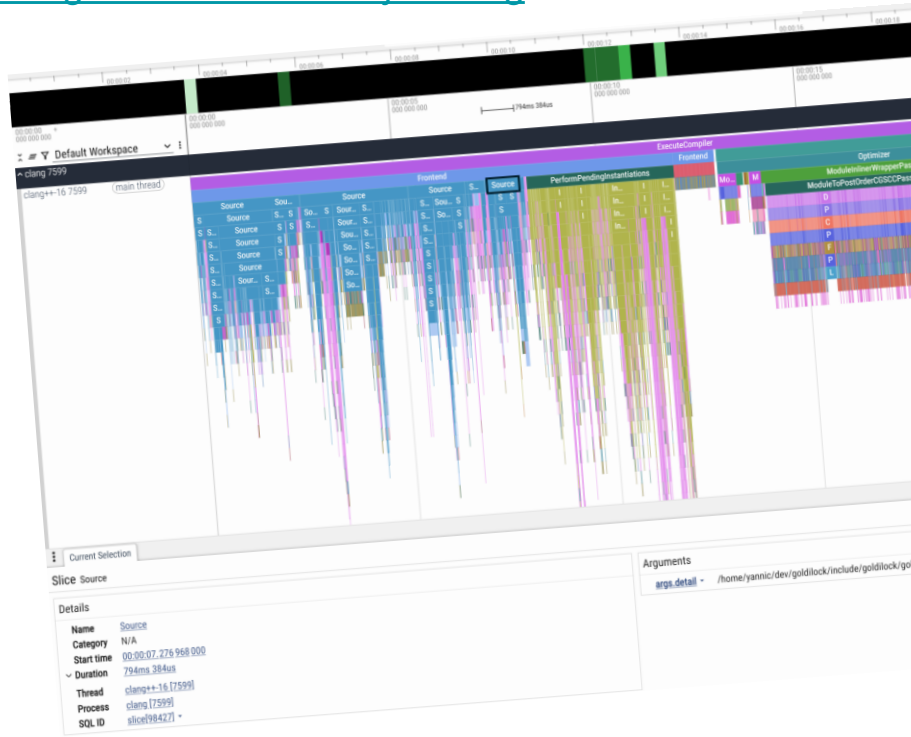
ONLY TOOLS AND HORSES

tipi.build
by EngFlow

When did you last profile your build?

- Clang users have it easy: `-ftime-trace=trace.json -ftime-trace-granularity=1`
- If you’re using ninja, there’s some great tools like https://github.com/nico/ninjatracing that use data from ninja’s build log
- For MSVC users, look into `vcperf`

- **Soon on our EngFlow platform dashboard!**



tipi.build by EngFlow

My binary is overweight. Who’s been feeding it biscuits??

- Give google’s `bloaty` a try it’s ***awesome(!)***
  https://github.com/google/bloaty

- A size profiler for binaries
- See what contributes most to binary size at the symbol, file, or section level.

- Shows where debug info bloats your binary, or what code/data is causing file size creep.

- Supports diffing so you can trace history and see impact vs. changes

```
    FILE SIZE        VM SIZE
 --------------  --------------
  47.9%  1.87Mi  76.0%  1.87Mi    .text
  12.3%   489Ki   0.0%       0    .strtab
  11.7%   468Ki   0.0%       0    .debug_info
   5.7%   226Ki   9.0%   226Ki    .eh_frame
   5.6%   223Ki   0.0%       0    .symtab
   4.4%   174Ki   6.9%   174Ki    .rodata
   3.3%   130Ki   0.0%       0    .debug_loc
   2.4%  95.0Ki   0.0%       0    .debug_line
   1.4%  57.8Ki   2.3%  57.8Ki    .data.rel.ro
   1.3%  50.8Ki   0.0%       0    .debug_abbrev
   0.0%     0     1.8%  44.6Ki    .bss
   1.0%  40.1Ki   1.6%  40.1Ki    .gcc_except_table
   1.0%  39.3Ki   1.6%  39.3Ki    .eh_frame_hdr
   0.6%  25.1Ki   0.0%       0    .debug_ranges
   0.6%  24.3Ki   0.0%       0    .debug_str
   0.2%  9.20Ki   0.4%  9.20Ki    .data
   0.2%  7.53Ki   0.3%  7.53Ki    __libc_freeres_fn
   0.2%  7.05Ki   0.0%       0    .note.stapsdt
   0.2%  6.36Ki   0.2%  5.56Ki    [25 Others]
   0.1%  2.69Ki   0.0%       0    [ELF Section Heade
   0.1%  2.53Ki   0.0%       0    .debug_aranges
 100.0%  3.90Mi 100.0%  2.46Mi    TOTAL
```

Level up your tooling game

(and maybe come over to talk with us – we have some cool stuff for CMake and Bazel [and much more] ;)