Jumbo

A unity build system for Chromium

Daniel Bratell

Opera

TL;DR

Jumbo: 3 times faster* builds now and possibly 9 times faster builds in the future.

*) Terms and conditions apply. Numbers for a full chrome+content_shell+blink_tests build on a 4 core/8 thread machine.

The problem

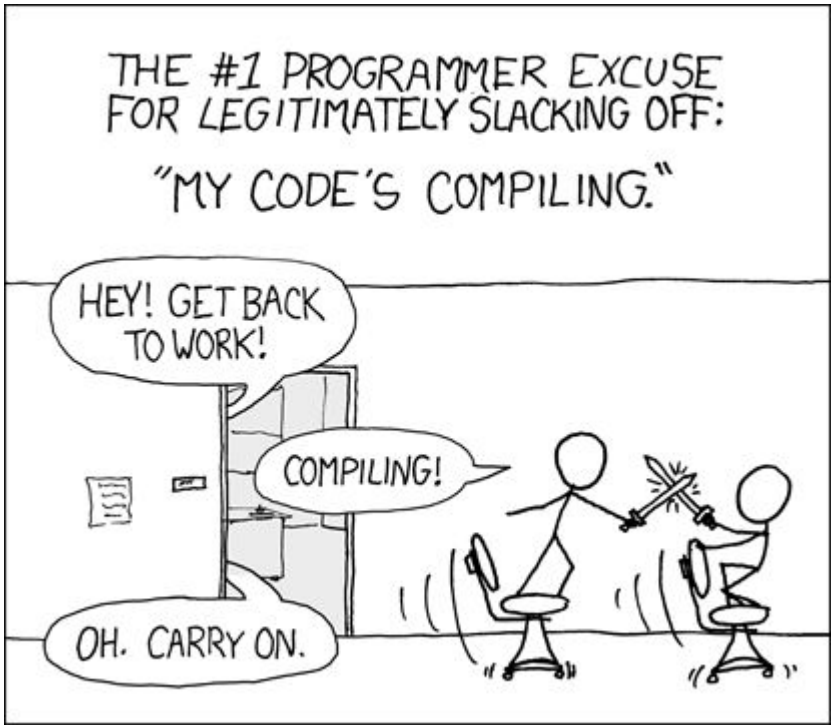
The compilation time for Chromium is long

Very long

Extremely long

Hours.

Opera



xkcd 303
https://www.xkcd.com/303/
(by Randall Munroe)

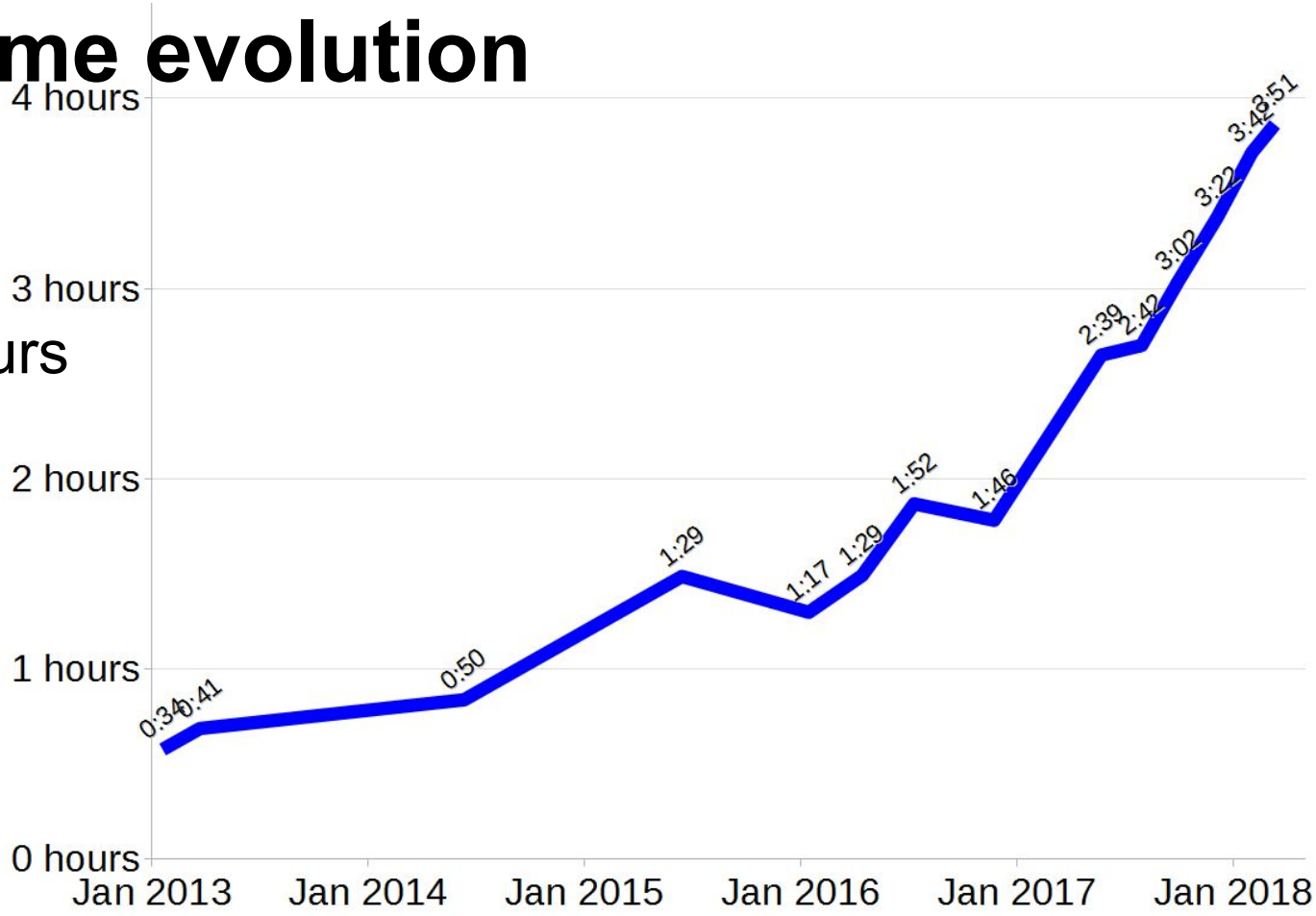
Opera

Compile time evolution

In five years

34 min → 4 hours

+1% per week

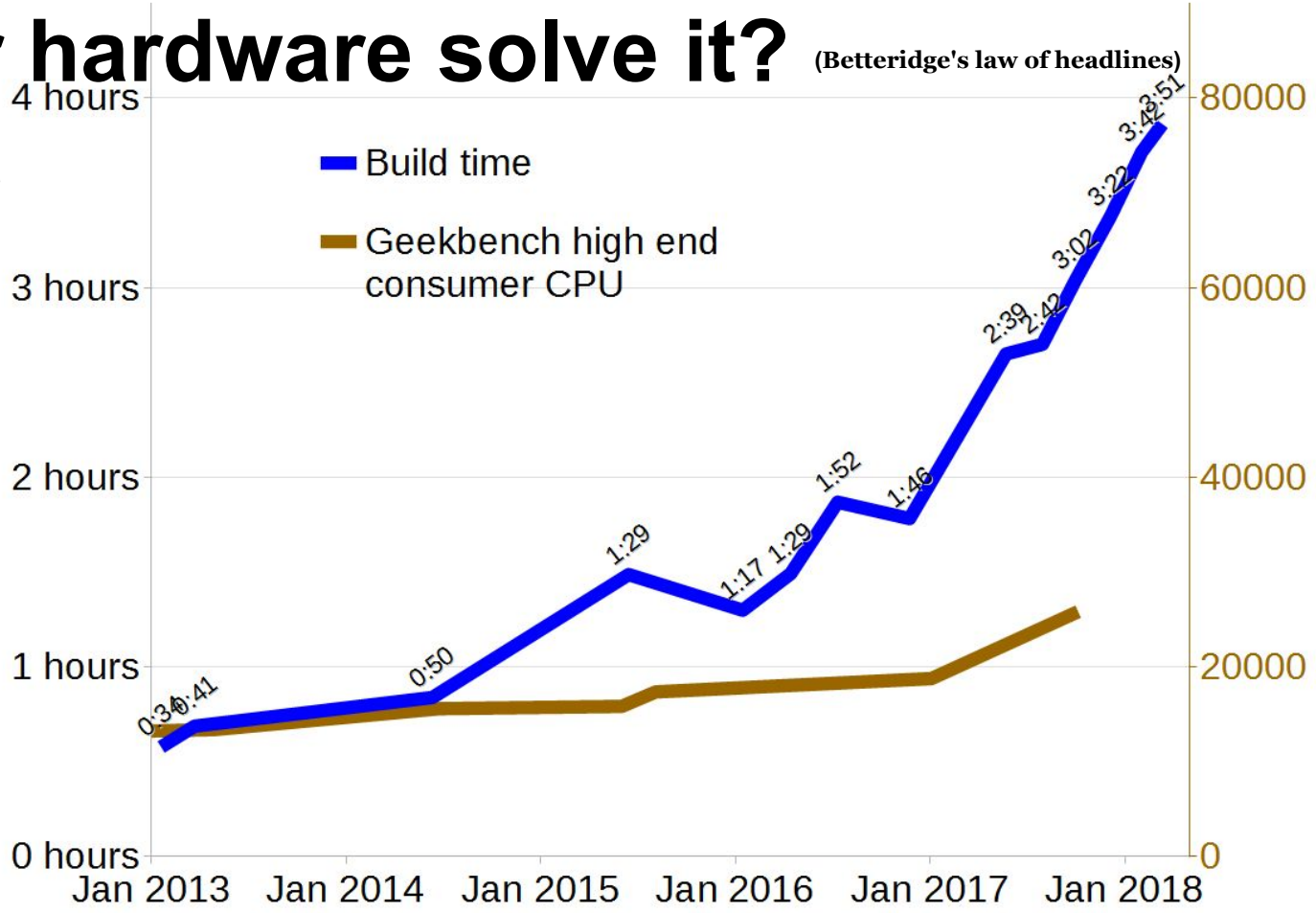


Opera

Will newer hardware solve it?

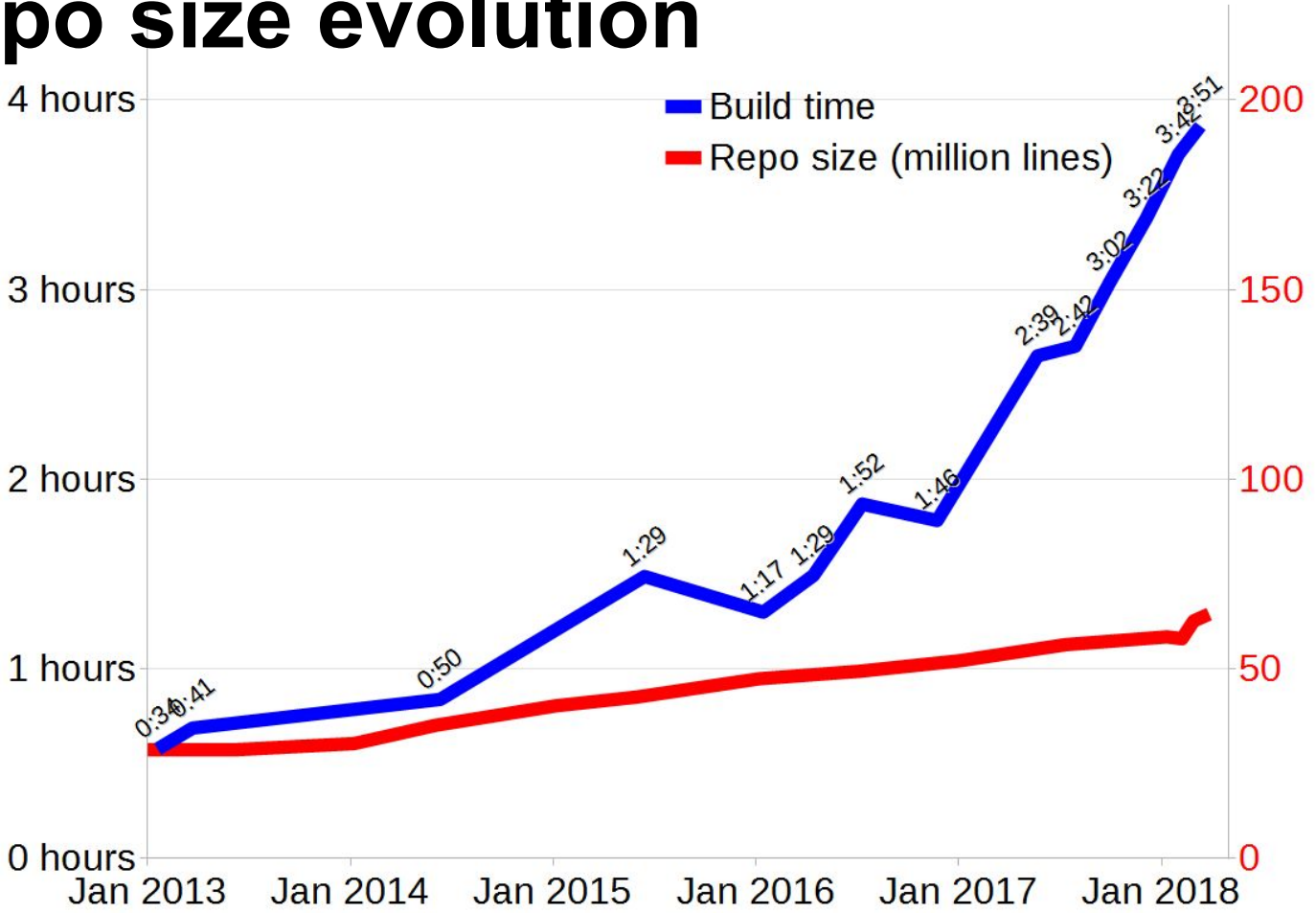
(Betteridge's law of headlines)

Hardware does not keep up



Opera

Time vs repo size evolution



Opera

Time consumers

- Code generation
- Compiling
- Linking

Can use the ninja log (`out/something/.ninja_log`)

No file needs more than 0.04% of the total time but:
There are 44,000 .cc files and 43,000 .cpp files in Chromium (including generated code).

Opera

Files are small

Median length `.cc` files is 134 lines
Median length `.cpp` files is 78 lines [pre-Blink move].

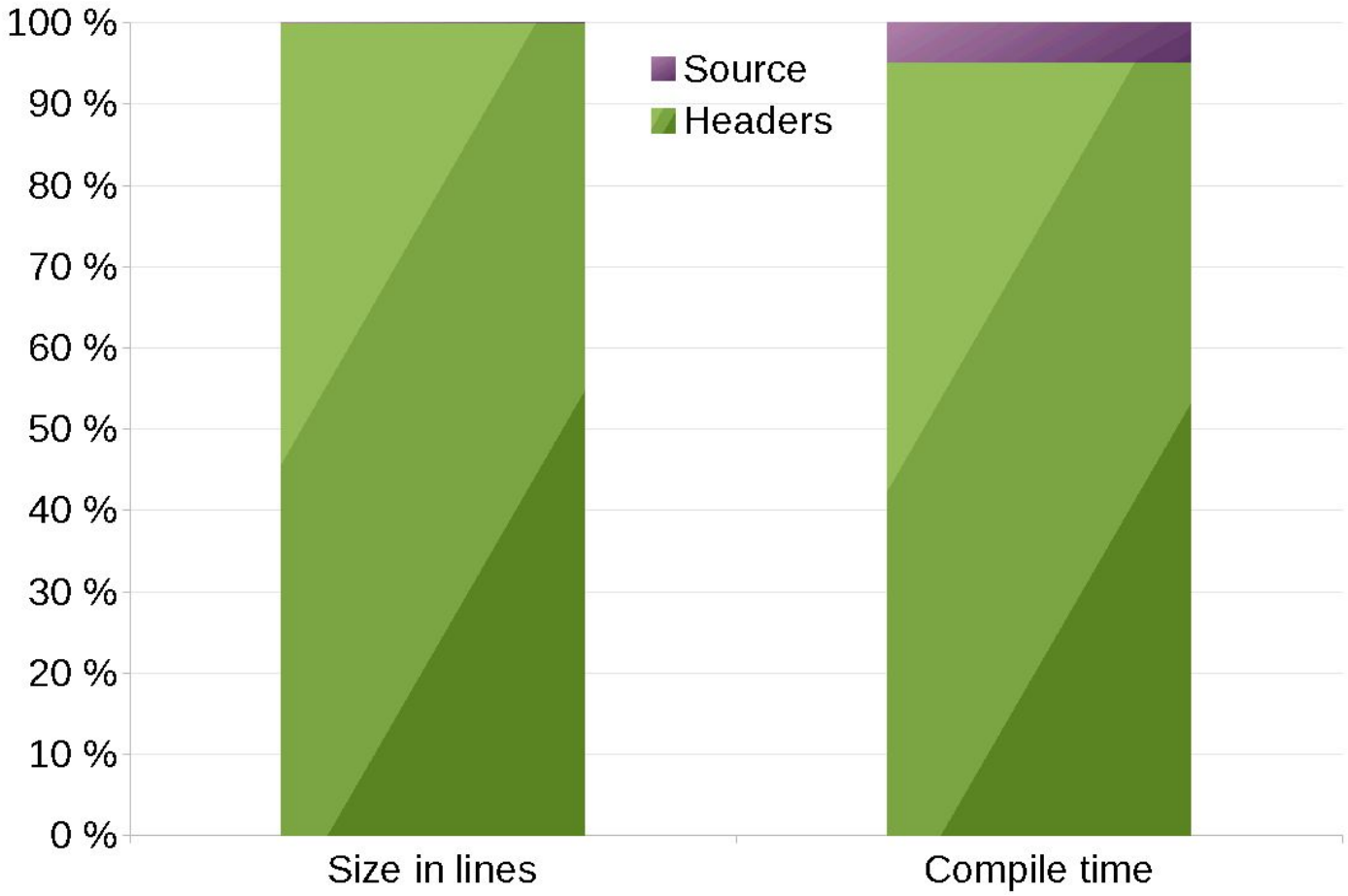
Preprocess a Blink file of length **110 lines** and you get **244,000 lines of source code**.

Opera

Per file

In lines the source file is nothing!

In compile time almost nothing



Opera

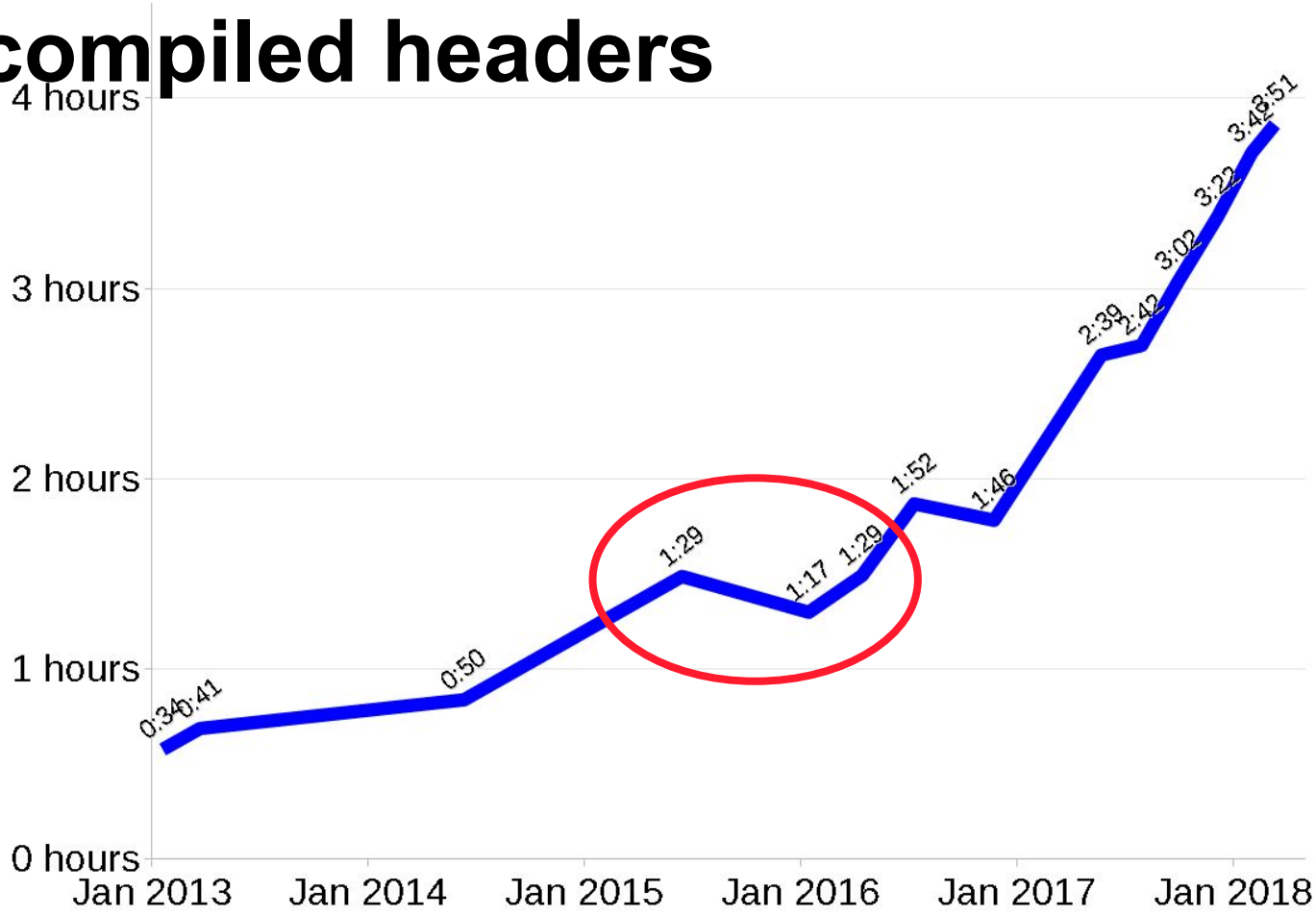
Precompiled headers

Been used for system headers for a long time

2015: Blink got more comprehensive precompiled headers

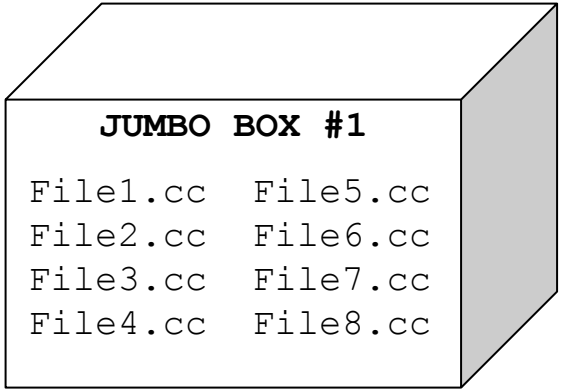
Saves 10-20% of the compile time

Effect precompiled headers



Opera

Unity builds



In Chromium: Jumbo builds

- Compile a lot of code in a single translation unit
- Common in large projects and the games industry
- Long used for blink's v8 bindings, and by third_party/sqlite
- Used in WebKit since February

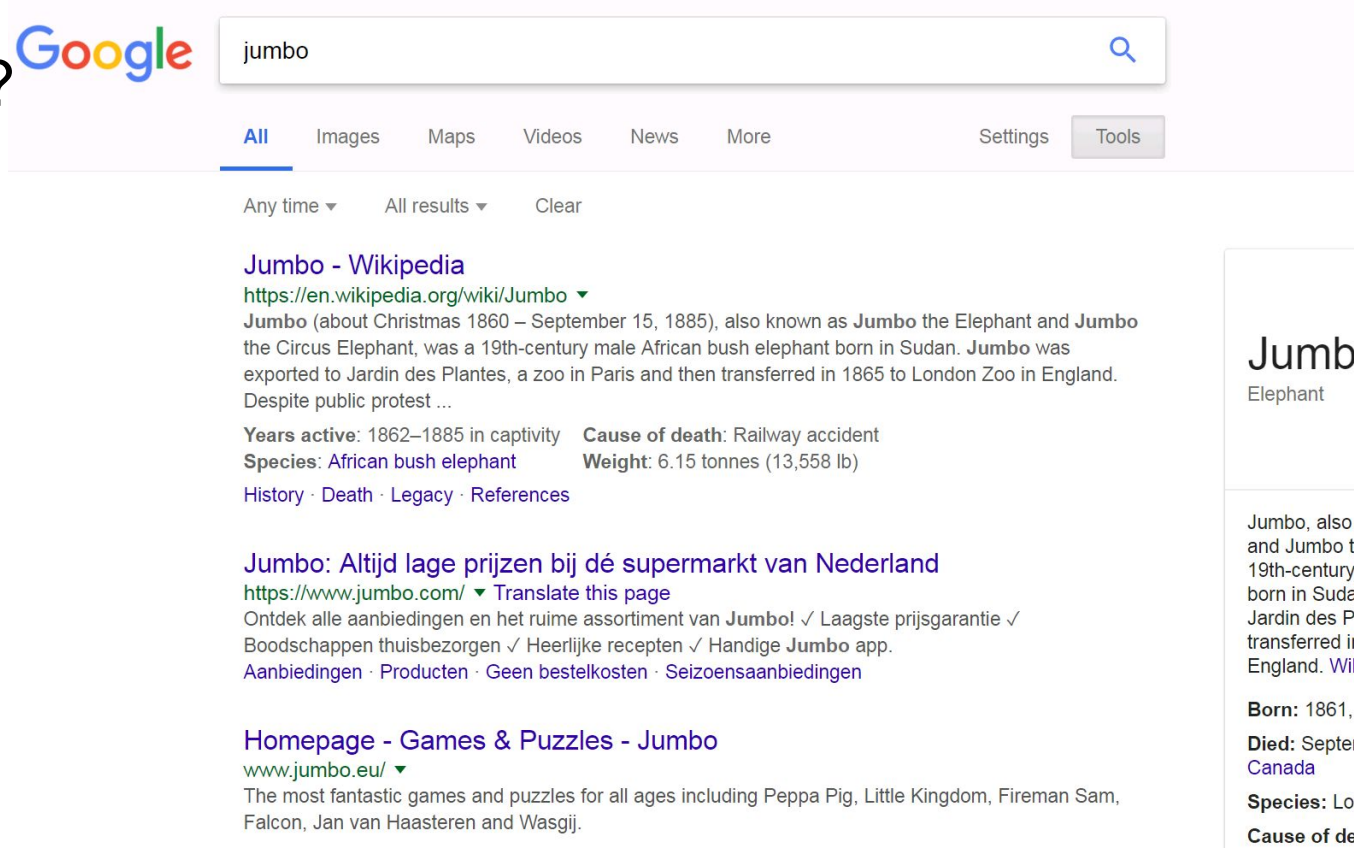
Opera

Jumbo

Is it an elephant?

Is it a dutch supermarket?

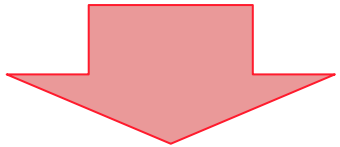
Is it a unity build system?



Opera

Jumbo in BUILD.gn

```
source_set("my_code") {
  sources = ["file1.cc", "file2.cc", "file3.cc"]
}
```



```
import("//build/config/jumbo.gni")
jumbo_source_set("my_code") {
  sources = [ "file1.cc", "file2.cc", "file3.cc"];
}
```

Opera

Jumbo in the file system

Template `action` generates `gen/…/`**`my_code_jumbo_1.cc:`**

```
#include "../../chrome/my_code/file1.cc"
#include "../../chrome/my_code/file2.cc"
#include "../../chrome/my_code/file3.cc"
#include "../../chrome/my_code/otherfile.cc"
```

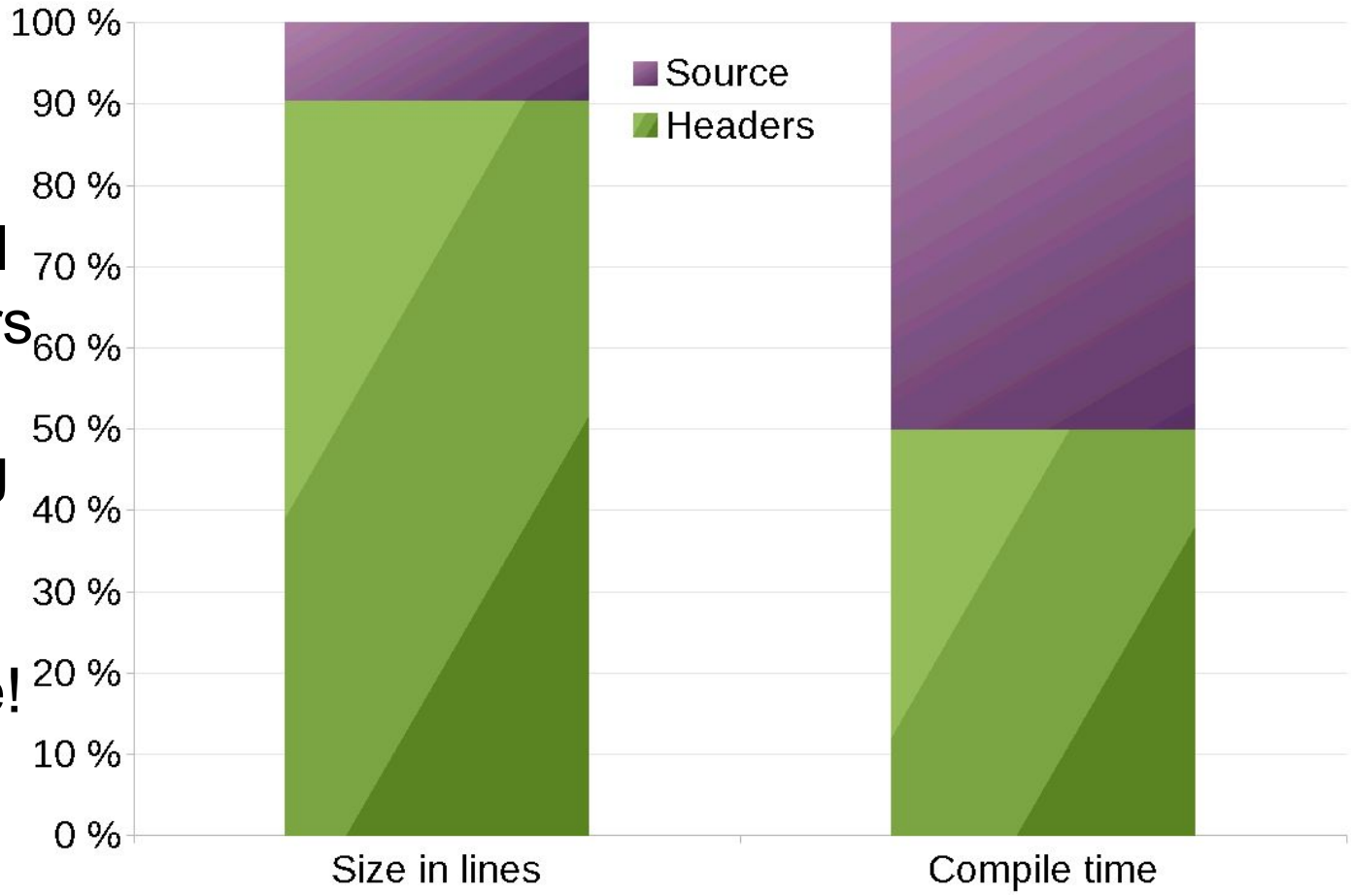
`my_code_jumbo_1.cc` compiled as usual

Opera

Jumbo

90% of the lines are still from headers

… compiling the source code is now half the time!

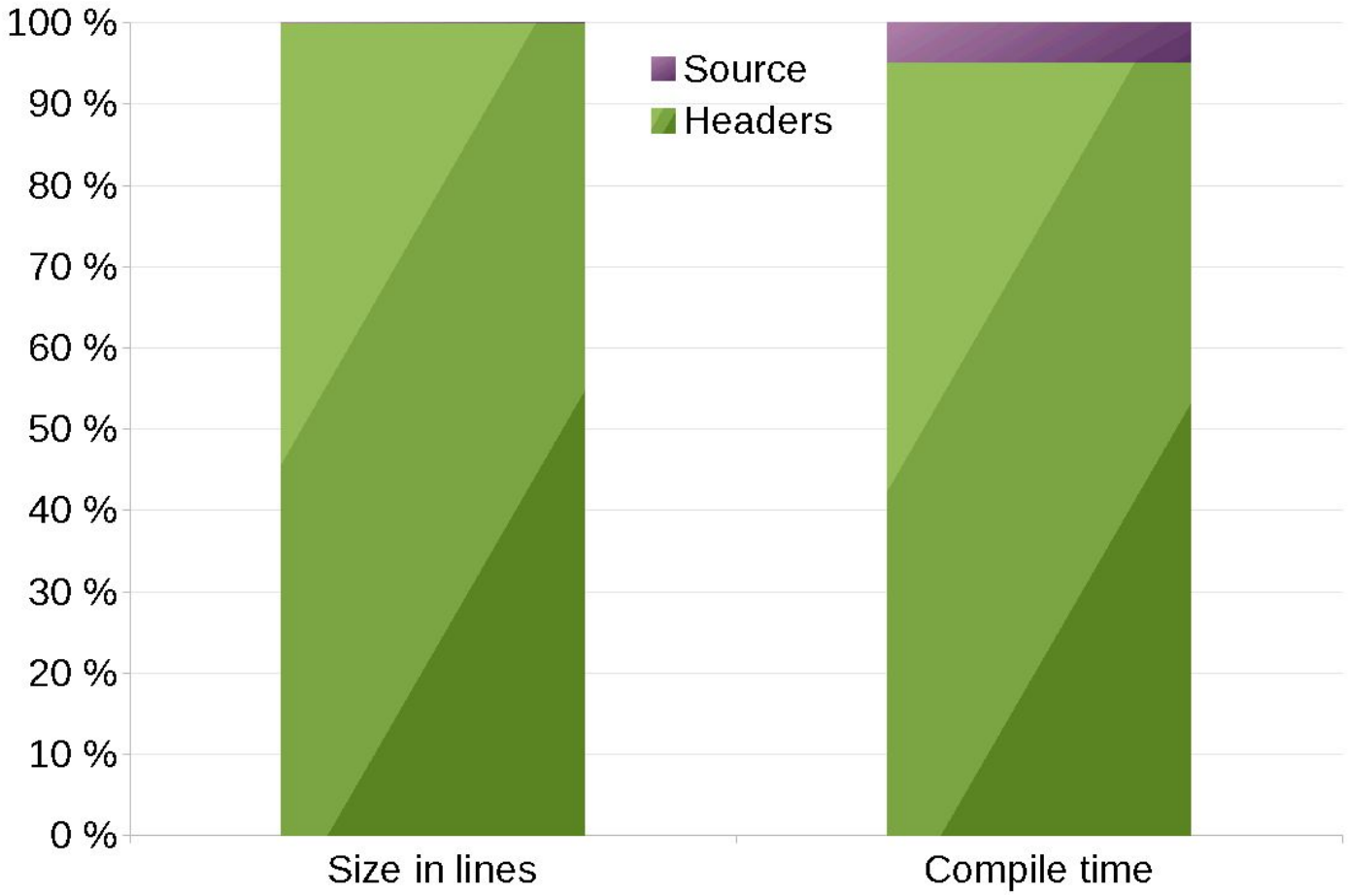


Opera

Per file

In lines the source file is nothing!

In compile time almost nothing

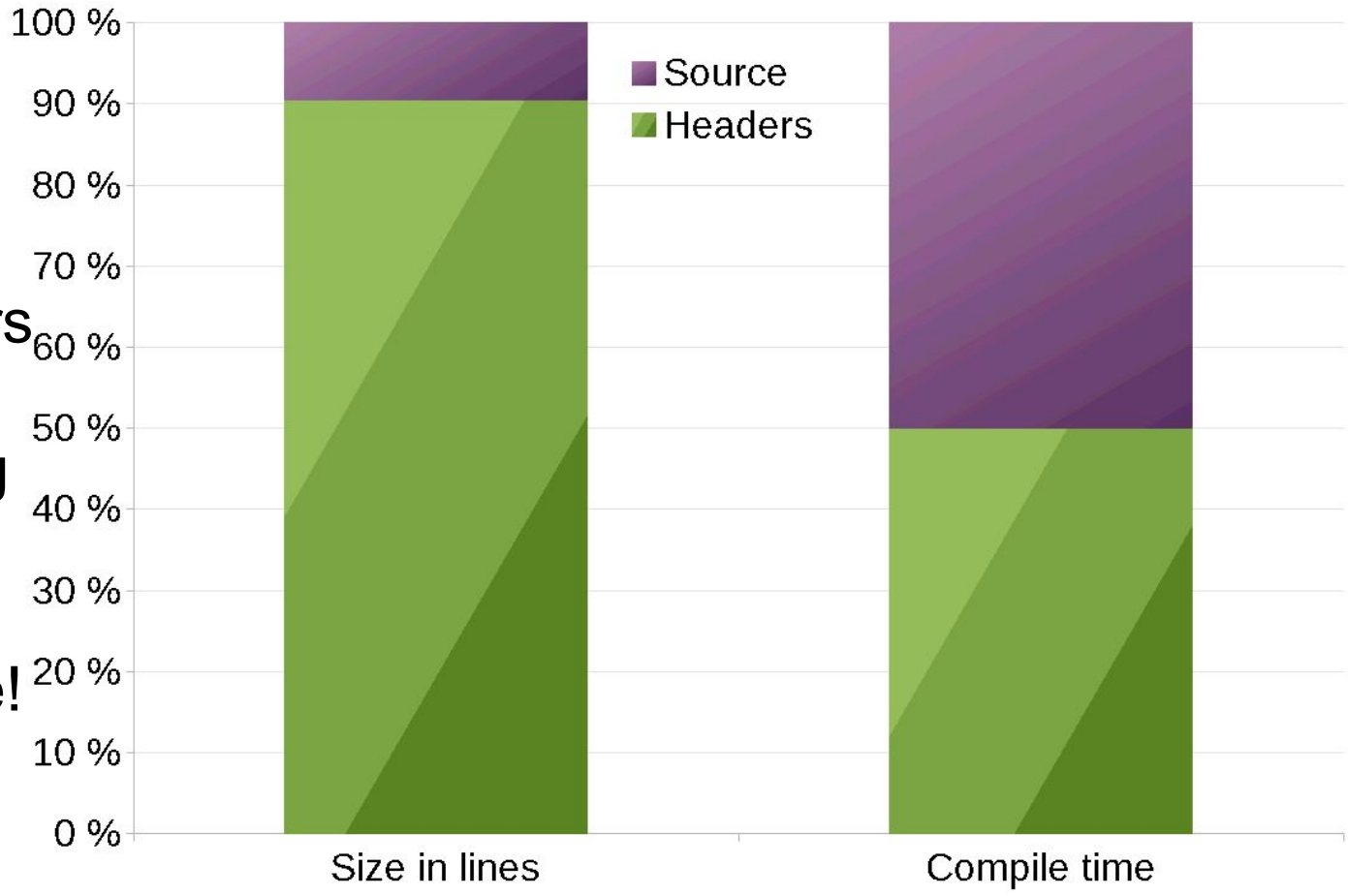


Opera

Jumbo

90% of the lines is still from headers

… compiling the source code is now half the time!



Opera

Metaphor

If you have a lot of cargo, some tools are more efficient

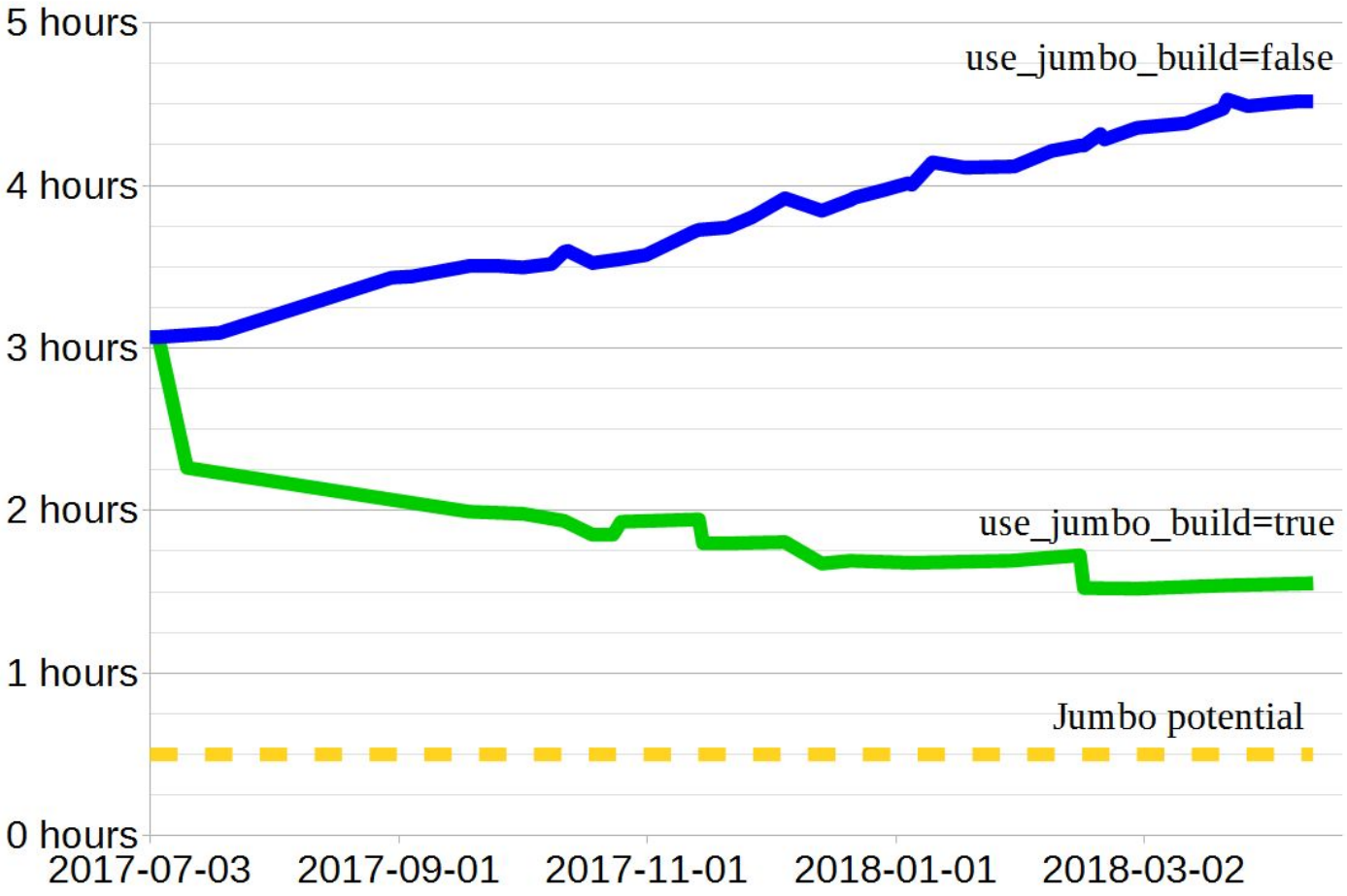
Metaphor

If you have a lot of cargo, some tools are more efficient

Jumbo

Compiling:
chrome+
content_shell+
blink_tests

4 core/8 thread

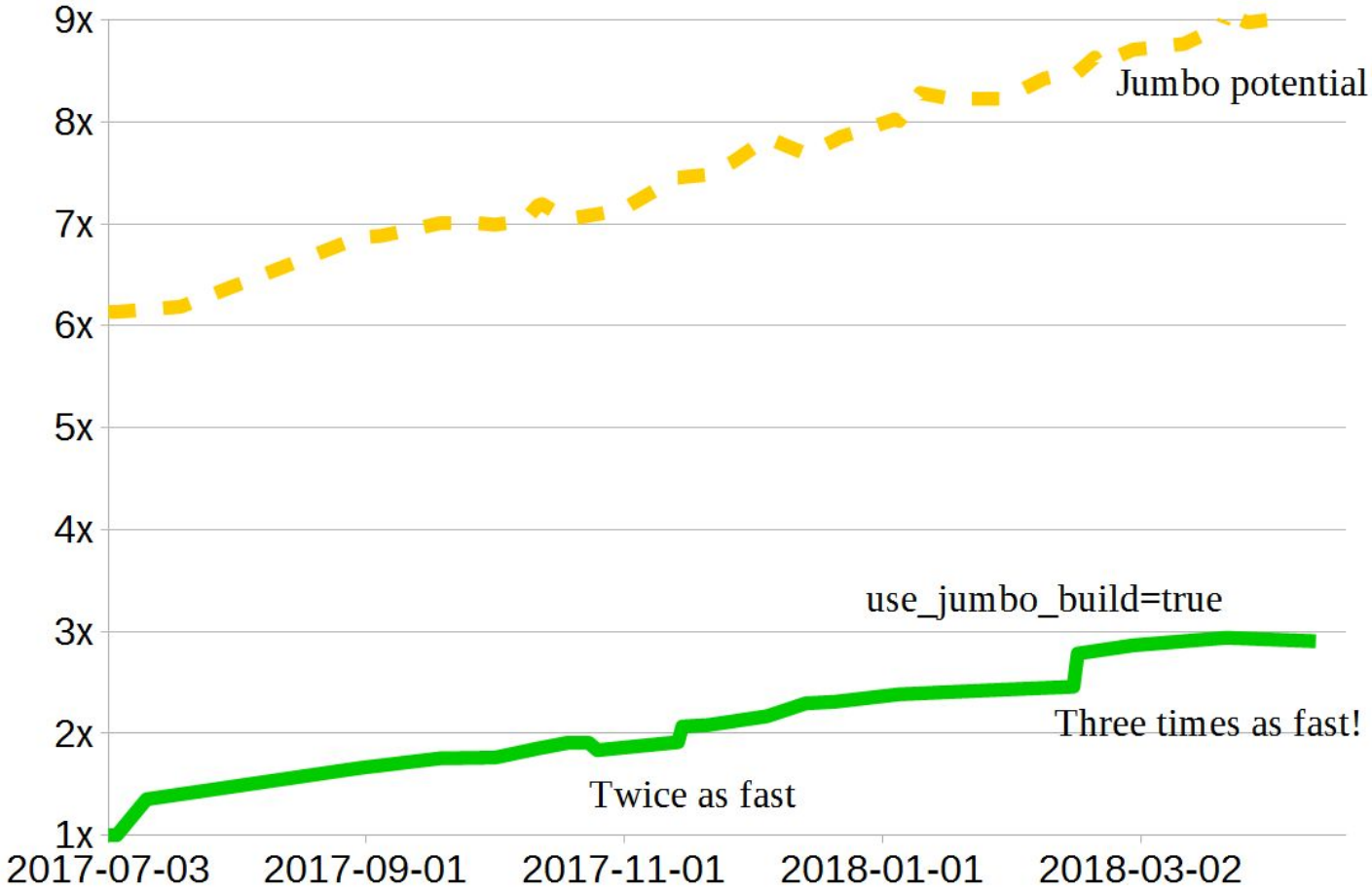


Opera

Jumbo

Compiling:
chrome+
content_shell+
blink_tests

On a 4 core/8
thread computer



Opera

Not always faster

- Always more efficient
  - Saves roughly 150 Wh per full build
- Less parallel
  - Concern if >100-1000 cores
    - Longer executing bottleneck tasks
    - Idling waiting for dependencies
  - Not an issue for "normal" hardware
- Tunable
  - Currently tuned for ~10-20 cores or ~100 cores if goma

Opera

Metaphor

If you have a lot of cargo, some tools are more efficient

Not always faster

- Always more efficient
  - Saves roughly 150 Wh per full build
- Less parallel
  - Concern if >100-1000 cores
    - Longer executing bottleneck tasks
    - Idling waiting for dependencies
  - Not an issue for "normal" hardware
- Tunable
  - Currently tuned for ~10-20 cores or ~100 cores if goma

Opera

Good things

Main target

- Average compilation time per file reduced by 80-95%

Accidental positive effects

- Less compiler output (debug component build tree 26 GB → 14 GB)
- Faster linking (due to less data to process)
- Cheap "global" optimization (per jumbo unit)

Opera

More positive side effects

- Duplicate code removal
- Dead code removal + dead member removal
- Addition of include guards
- Solved the X11 header problem (no more `#undef None`)
- Namespace renamings, ~~no more~~ fewer `::`-prefixes
- Easier to find symbols when their names are more distinct than `content::{anonymous namespace}::kValue`

Opera

Complications

- With jumbo: "1 cc file" ≠ "1 translation unit"
  - local names become less local.

To a lesser degree:

- More code is exposed to system headers
- Stress testing tools ~~("objcopy O(n^2) in number of sections")~~
- Worse IDE support in gn
- Increased single file rebuild times (unless linking makes up for it)

Opera

Infrastructure

3 fyi bots

Mail a selected few when one of them break (sometimes twice a day, sometimes once in a week)

CQ support coming any day/week

Opera

To do

CQ support

- Star crbug.com/782863 for updates (unless it's already fixed)

Add support to the rest of Chromium (after CQ)

- Tests, components, services, third_party

Native gn support

- For better IDE support
- PoC/patch by Tomasz Moniuszko exists
- Star crbug.com/772918 for updates

Opera

Assistance by clang (PoC covered in lightning talk)

Create sandboxes for each file and prevent some of the problems

PoC: Add pragma that hides/disables current anonymous namespaces. Can not be developed purely as a plugin.

Doesn't solve all problems but prevents the more annoying ones

Time frame: Uncertain, months to infinity

Deprecating other systems

- split_static_library
  - Needed because libs > 2 GB. Not the case in jumbo builds
- component builds
  - Solving long link times. Linking is (a bit) faster in jumbo
- Putting a lot of source in one file
  - v8 has not split up its code because of compile times. Faster in jumbo?

Opera

Jumbo alternatives

Distributed compilation (icecc, distcc, an open and independent goma)

- Can sometimes be combined with jumbo for win-win

Faster compilers

- clang can be 20% faster if compiled with other flags
- compilers that cache state
- C++ modules?
- Can be combined with jumbo for win-win

Another implementation language than C++?

Open source goma instead of clang

goma client open source while server still proprietary

If depends on Google's infrastructure
- Intellectual property
- Price
- Dependence
- "Privacy"

Else
- Access to hardware (works for large companies but not individuals and small companies)
- Maintenance

How to make it faster?

Less code
- No

Precompiled headers
- Saves only 10-20%

Faster compilers
- No silver bullet, maybe 20%

Faster hardware
- Not fast enough

Distributed compilation
- Really helpful **if …**
  - Massive hardware available
  - Distribution systems work
- Can save 90+% of the time

Unity builds
- **Requires code changes**
- **Can save 90% of the time**

Opera

Questions

"If you are wondering something, there are probably many others that would like to know the answer as well."

/ Me - right now

Opera

Thanks!

Daniel Bratell @ Opera
Mostyn Bramley-Moore @ Vewd

Bruce Dawson @ Google	(Windows/goma)
Dirk Pranke @ Google	(infrastructure)
Tomasz Moniuszko @ Opera	(native gn, IDE support)
Jens Widell @ Opera	(clang support)

And many, many others (haraken, thakis, the_stig, pdr, fs, avi, kinuko, sky, sadrul, ...) who have tested, experimented, reviewed and added jumbo support to code