-Os Matters

Mark Zeren

C++Now, May 10, 2018

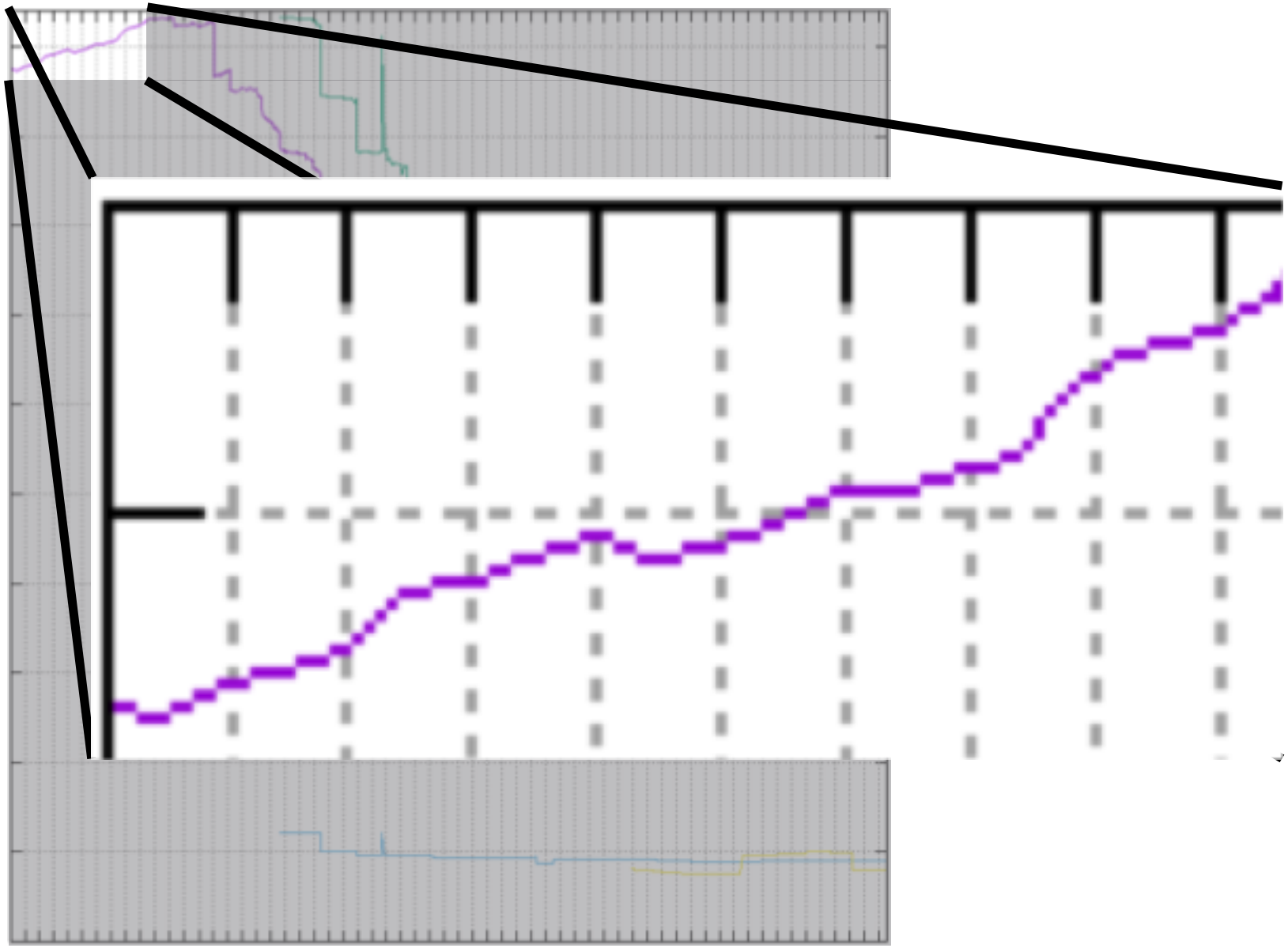
vmware

(Credit to **Ole Agesen**)

Fast Growth Company

- We add features, ergo
- We add code.
- We under-allocate tech debt cleanup, ergo
- We never remove code.
- (It’s C++. It’s big!)

Growth



We don’t see the graph

- Code is brittle
- Code is slow
- Code is hard to understand

Contributors

- Add a feature
- Duck

Everyone has seen this

Mature Product

- Features being added elsewhere in the company
- The most important feature we can provide is to do what we are doing, just better.

Complexity

Mature Product

- Motivated but,
  - Many lines of attack
  - Many opinions
- Requires
  - Staffing
  - Leadership
  - Workflow

The Herding Problem

A classic problem in
software engineering

Metrics

How do we collectively make forward progress?

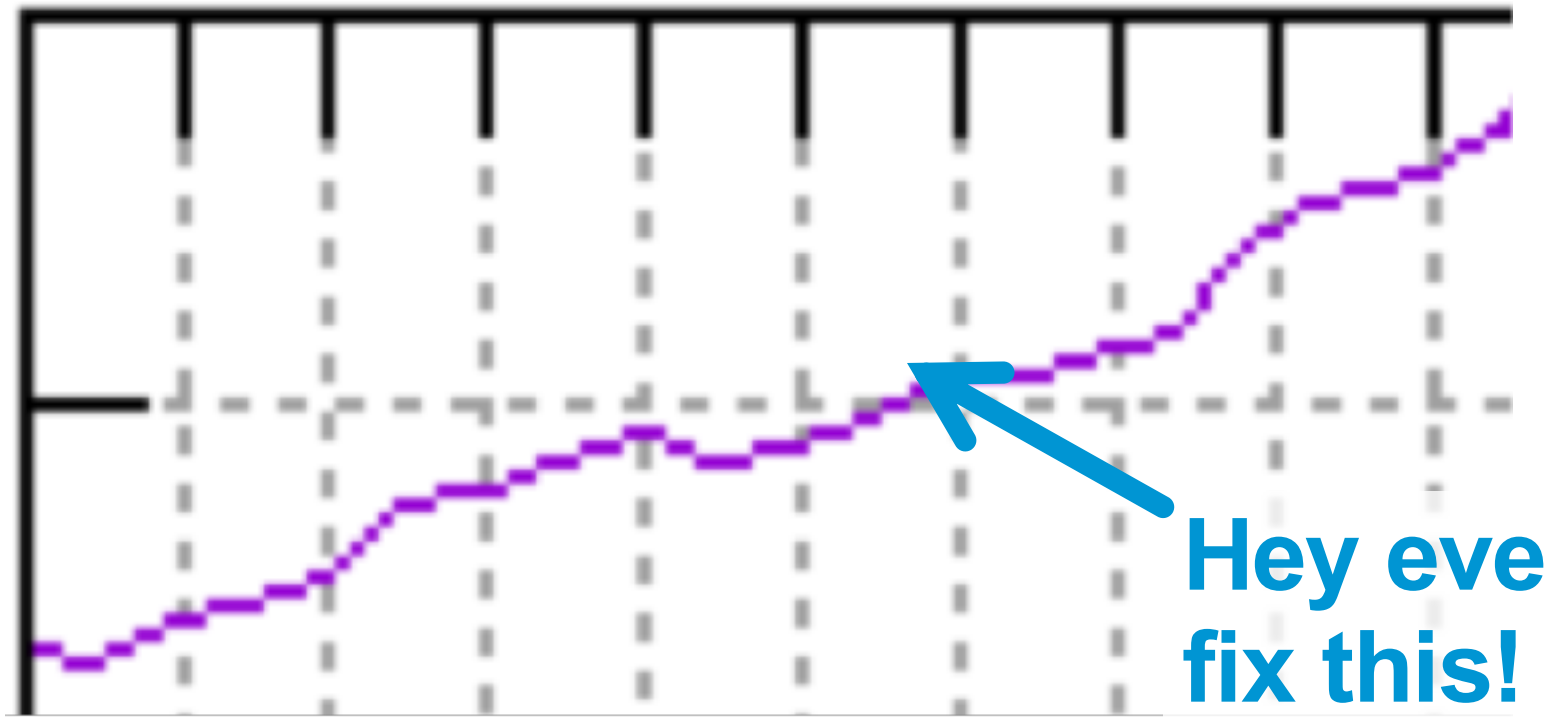
Of course we measure:

- Ops / second
- Resources / op
- Max capacity
- Uptime
- Micro-benchmarks
- Coverage…

But…

- Do not drive decreases in *complexity*
- (Except maybe coverage)
- Some drive increases in complexity
- Latency of integration tests
- How to incentivize complexity reduction
- At every commit?

Enter: Binary size



mylib: Remove temporaries

Remove unnecessary string copies.

Testing Done:
* Unit tests.
* mylib size:

```
      text       data        bss        dec   filename
    194527      21900      74873     291300   mylib-after
    194647      21900      74873     291420   mylib-before
      -120          0          0       -120 bytes
```

```
bash-4.1$ ls -ld ../src-*
drwxr-xr-x 19 mzeren mts  4096 Apr 24 13:17 ../src-base
drwxr-xr-x 18 mzeren mts  4096 Apr  3 18:21 ../src-this
drwxr-xr-x 18 mzeren mts  4096 Apr  5 18:14 ../src-that
bash-4.1$ ls -ld ../bld-*
drwxr-xr-x 19 mzeren mts  4096 Apr 25 20:28 ../bld-base
drwxr-xr-x 18 mzeren mts  4096 Apr 25 21:15 ../bld-this
drwxr-xr-x 18 mzeren mts  4096 Apr 25 21:34 ../bld-that
bash-4.1$
```

```
bash-4.1$ size mylib.so
   text    data     bss     dec     hex filename
 194527   21900    74873  74873   12479 mylib.so

bash-4.1$ src-base/scripts/mylib-size bld-this bld-base
 mylib size:
      text       data        bss        dec   filename
    194527      21900      74873     291300   mylib-after
    194647      21900      74873     291420   mylib-before
      -120          0          0       -120 bytes
bash-4.1$
```

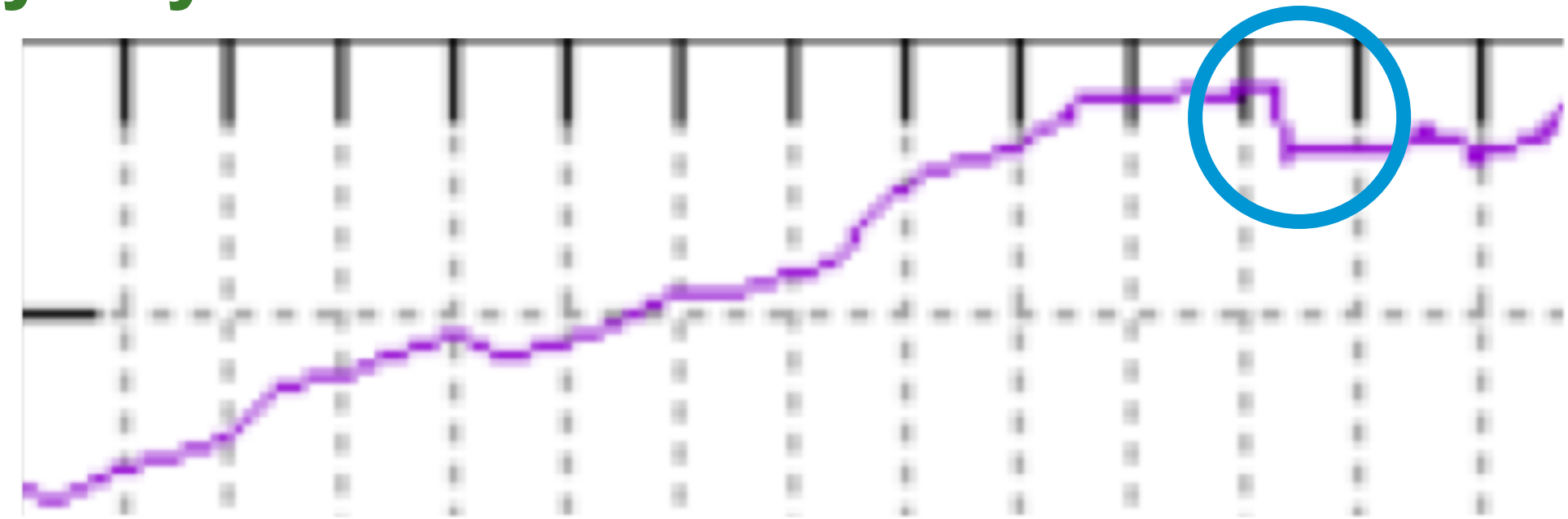
- Part of your edit compile loop
- Copy-n-paste into review description

“You’ve got to be kidding”

No, not really…

- People keep multiple projects anyway.
- You want to run locally
- It's like a unit test
- Did I pass my test?
- Did I bloat my app?
- It is part of self-review
- It is part of design

Early Days



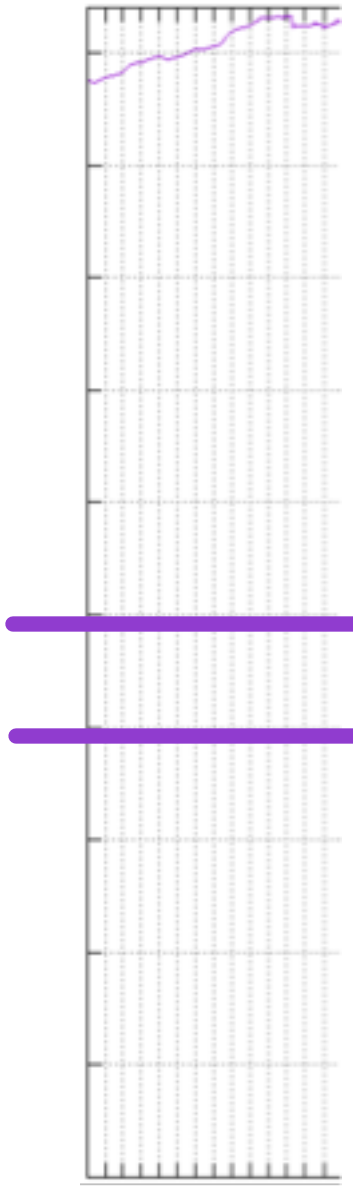
Manual de-virtualization, base class simplification

OK, but context please?

OK, but context please?

1%
gcc 4.3 -O2
Release
Stripped
Nightly
Years

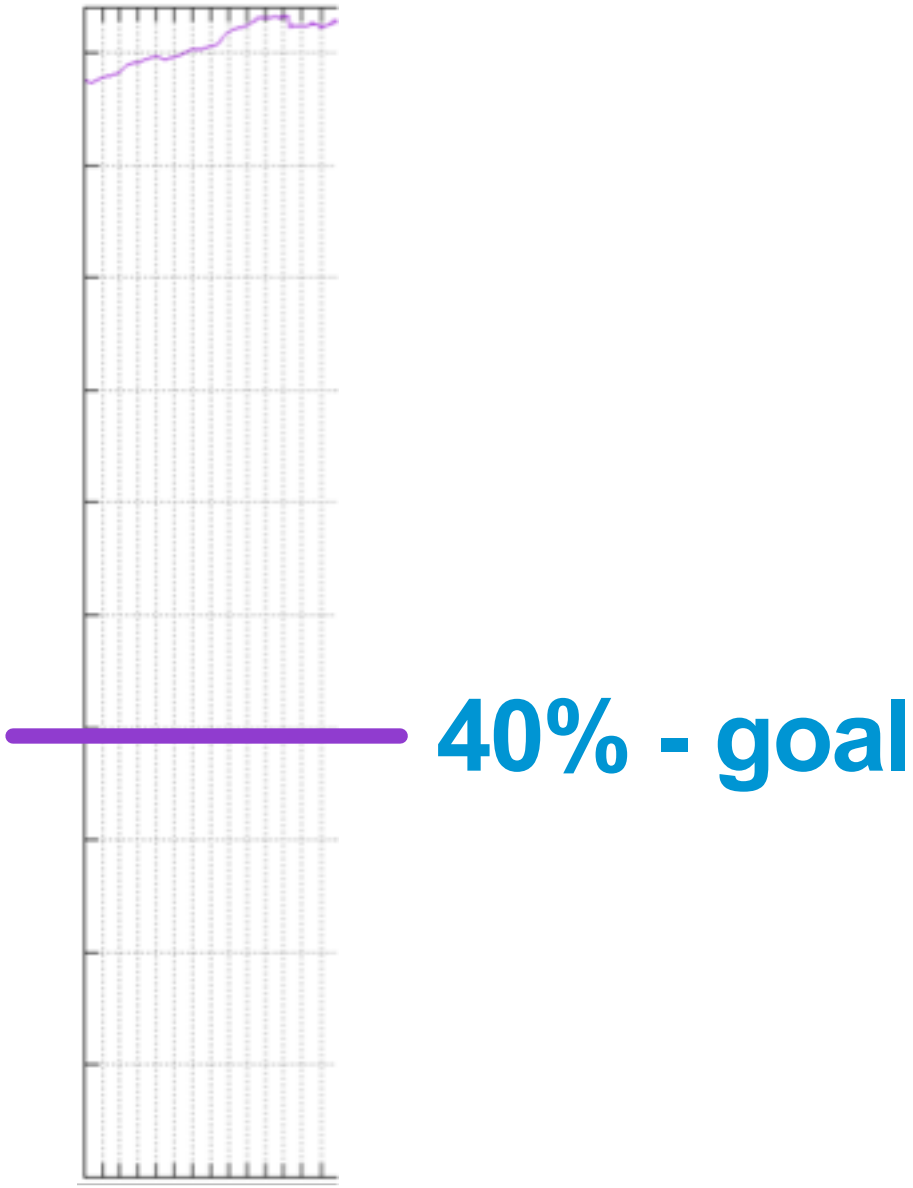
OK, how far can we go?



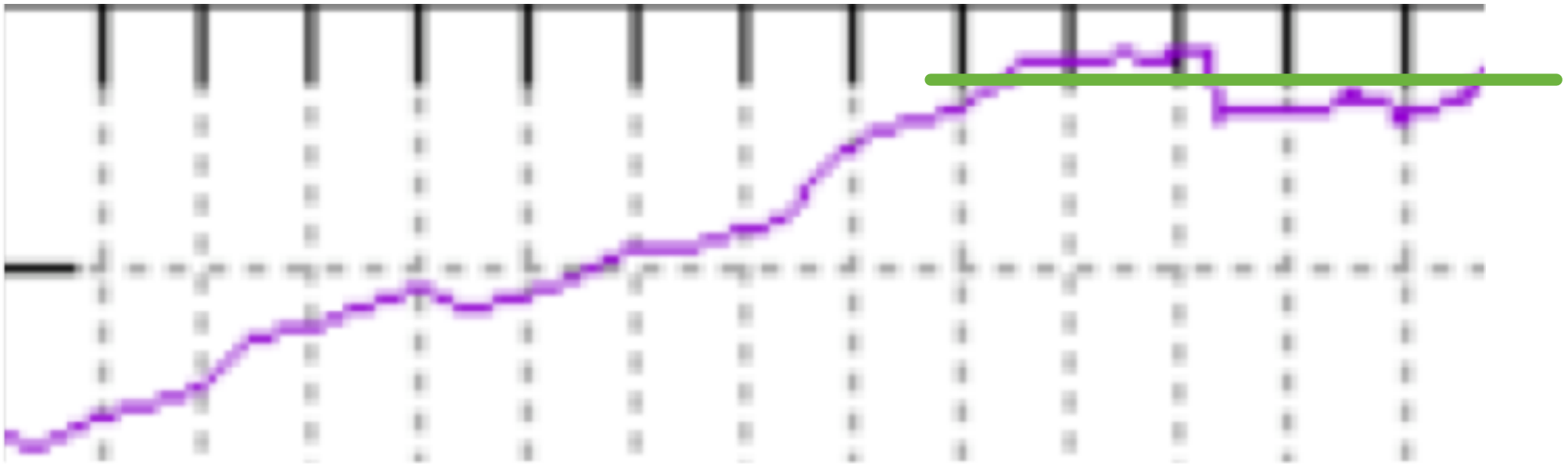
50% - goal?

40% lower bound

OK, how far can we go?

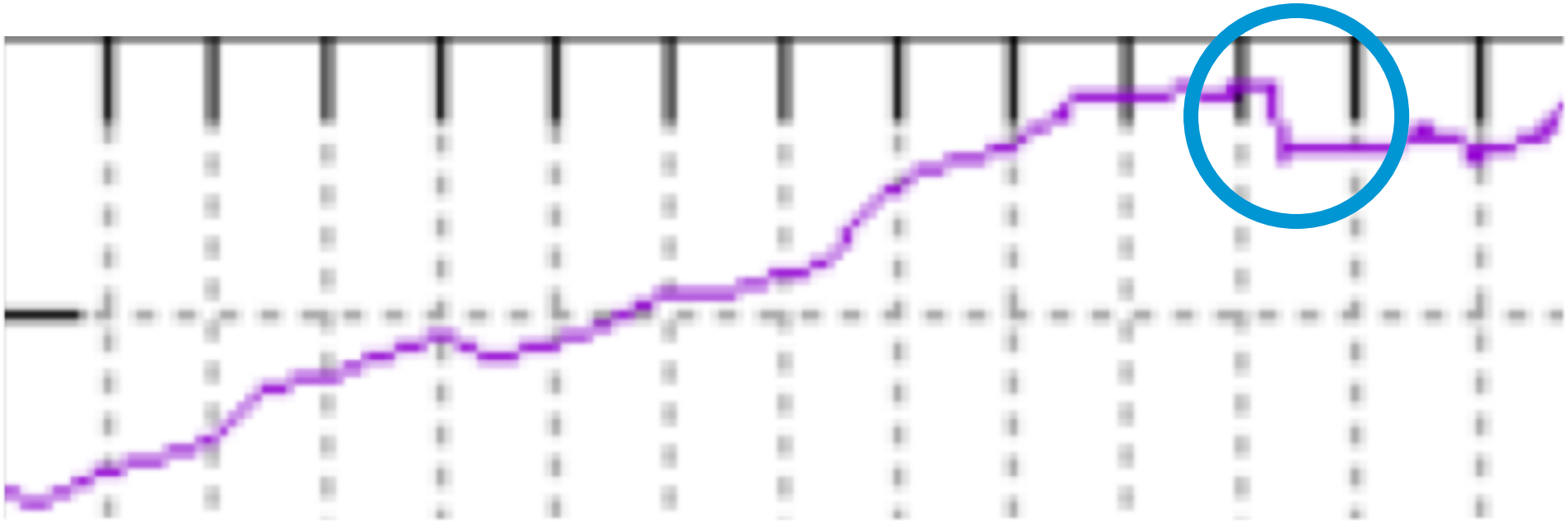


What else do you see here?



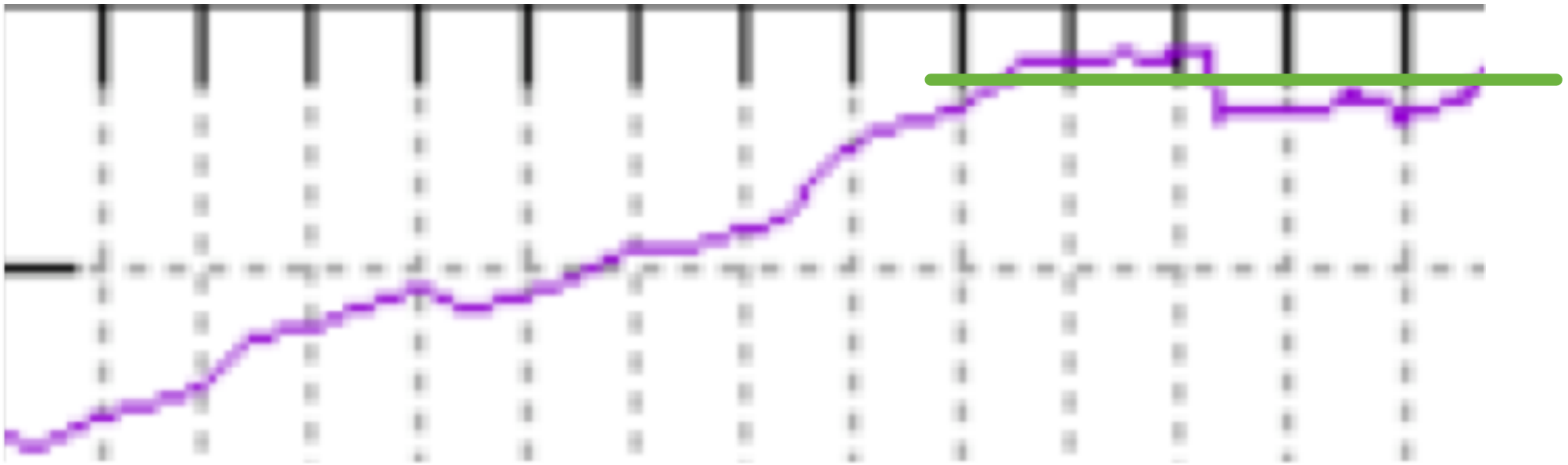
Ole

Tools



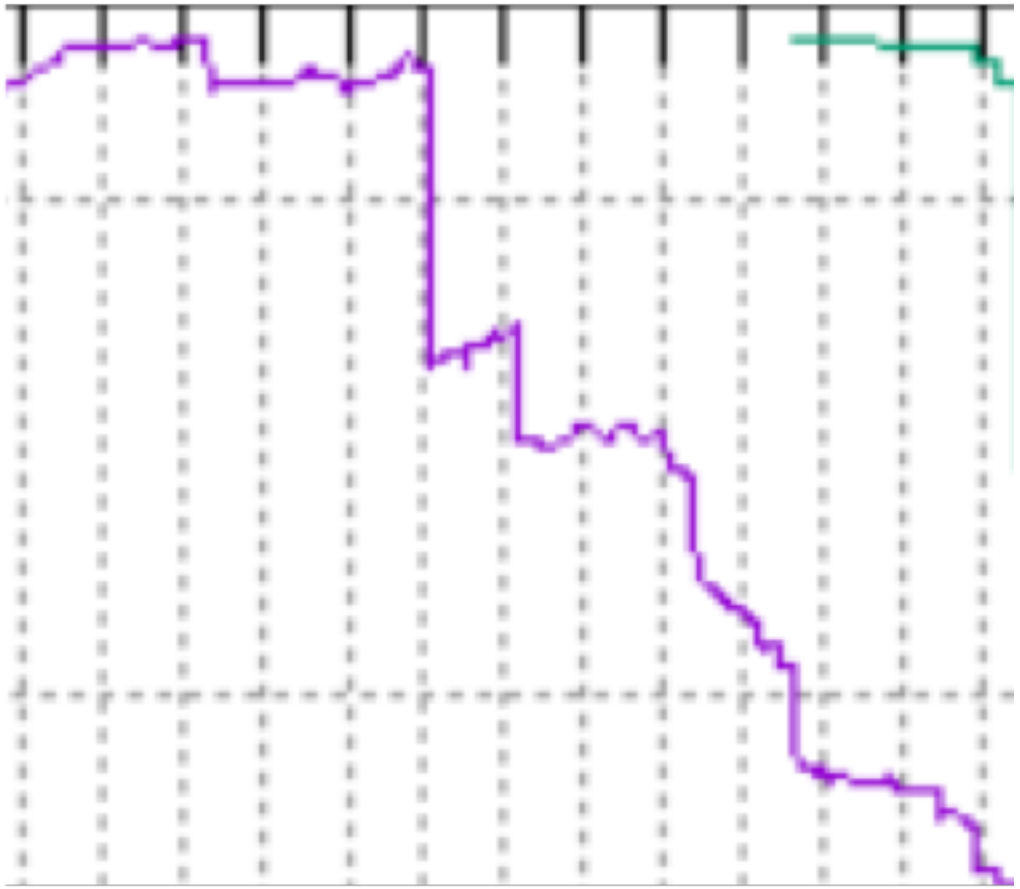
- Libraries, emitted code, compilers, etc.
- Percentages: 0.5%, 1%, 5%
- Discontinuities

Humans

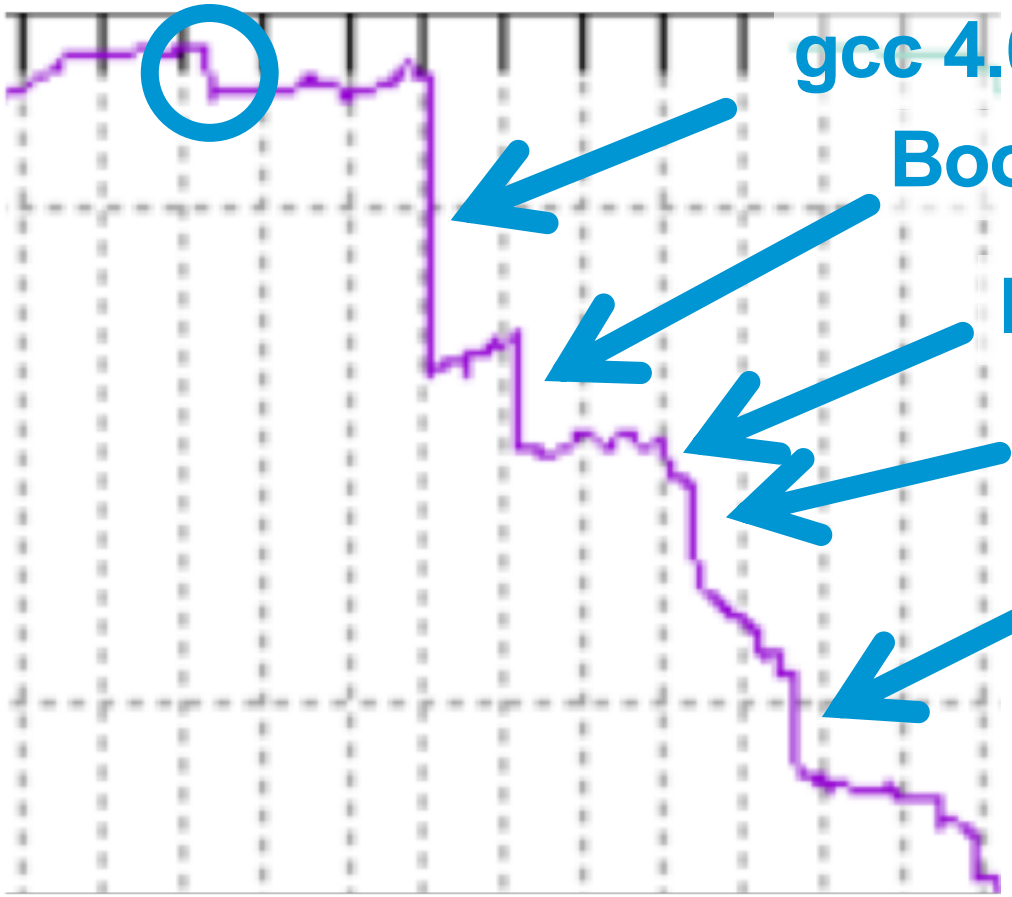


- Step by step, easy, small changes
- Absolute numbers: 150 bytes, 1K, 20K, 0K

Gaining Momentum …



Gaining Momentum



gcc 4.6

Boost disable asserts

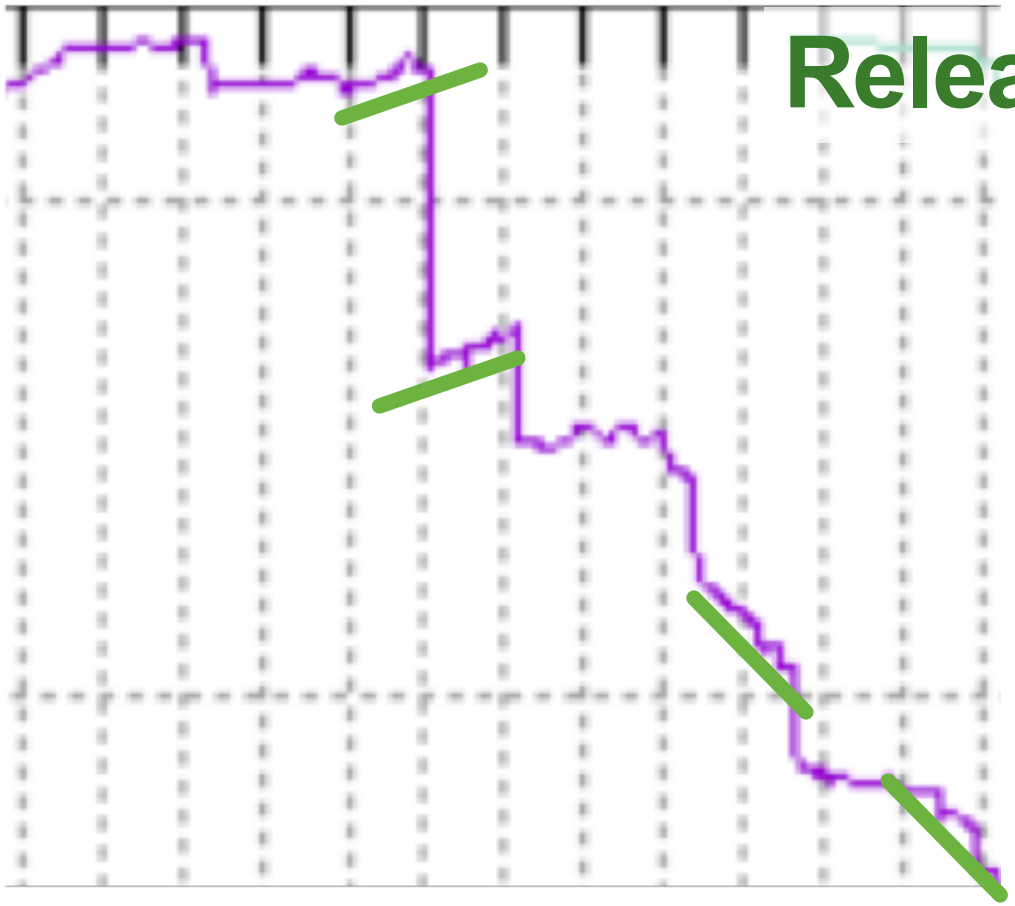
Log formatters

`extern const std::string`

Removed 3rd party library

Removed service reference

Gaining Momentum



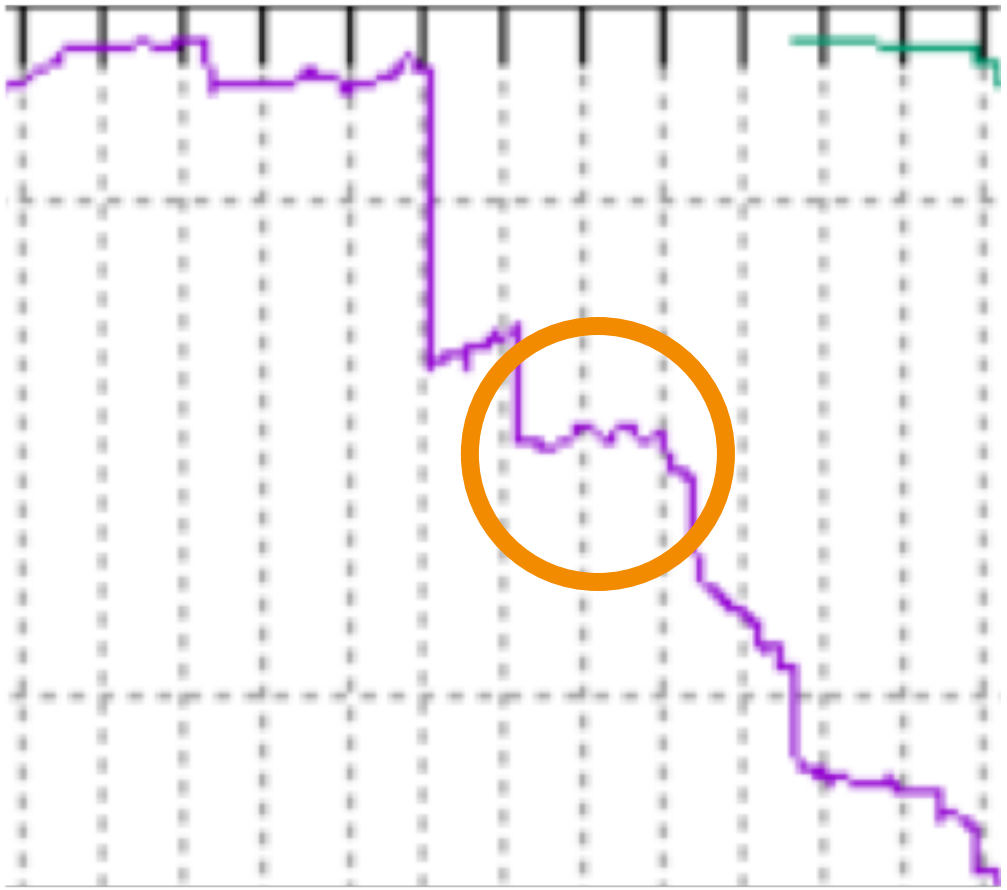
Release crunch

Motivated team

```
mylib: Remove string copies

* Switch to const& locals to avoid string copies.
* Other no-op cleanup.

Testing Done:
* Unit tests
* mylib size:
      text      data       bss       dec   filename
    194527     21900     74873    291300   mylib-after
    194647     21900     74873    291420   mylib-before
      -120         0         0      -120 bytes
```

```
mylib: Remove more string copies, etc.

* More const& locals.
* More no-op cleanup.

Testing Done:
* Unit tests
* mylib size:
      text       data        bss        dec   filename
    194757      21900      74873     291530   mylib-after
    194647      21900      74873     291420   mylib-before
       110          0          0        110 bytes
```

What is my change actually doing?



Whatever the inliner feels like doing today

the big hammer

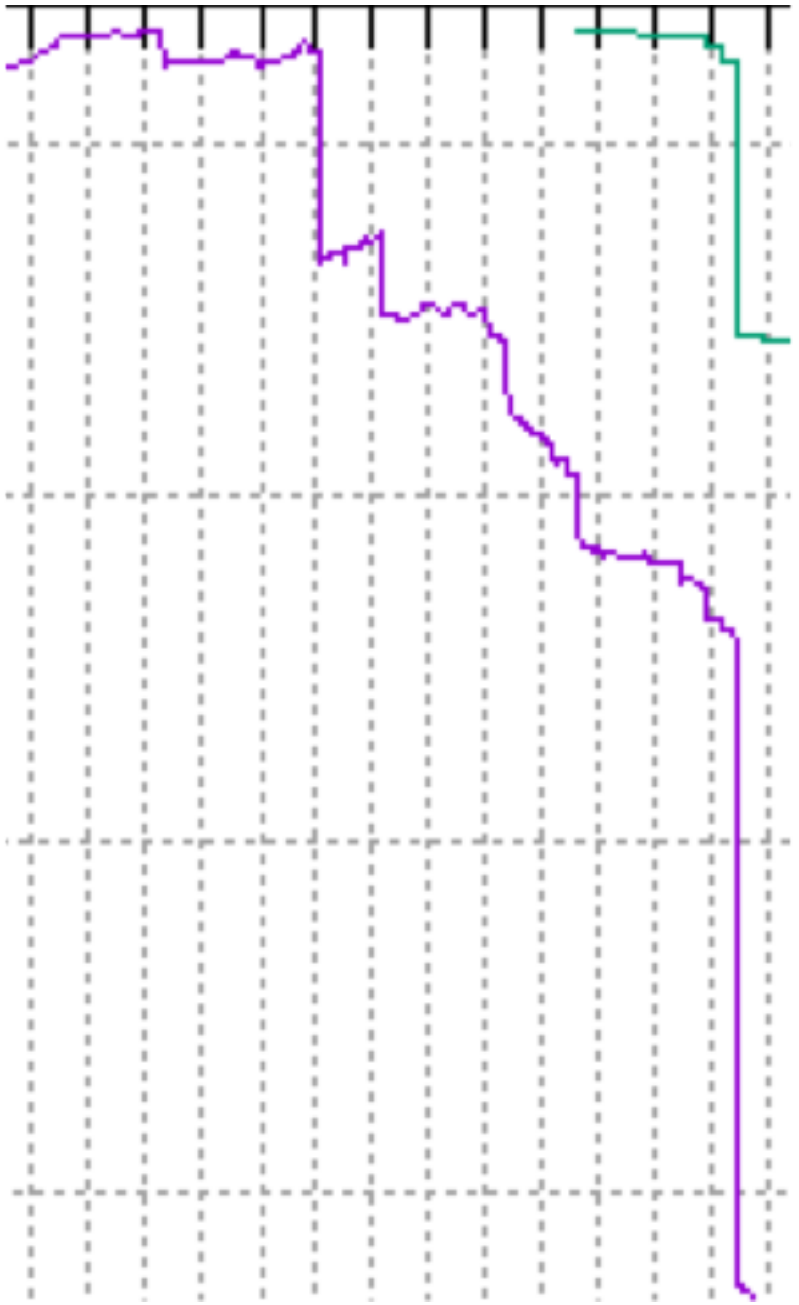
-Os

the big hammer

-Os

- ~~the big hammer~~
+ the small hammer

The small hammer…



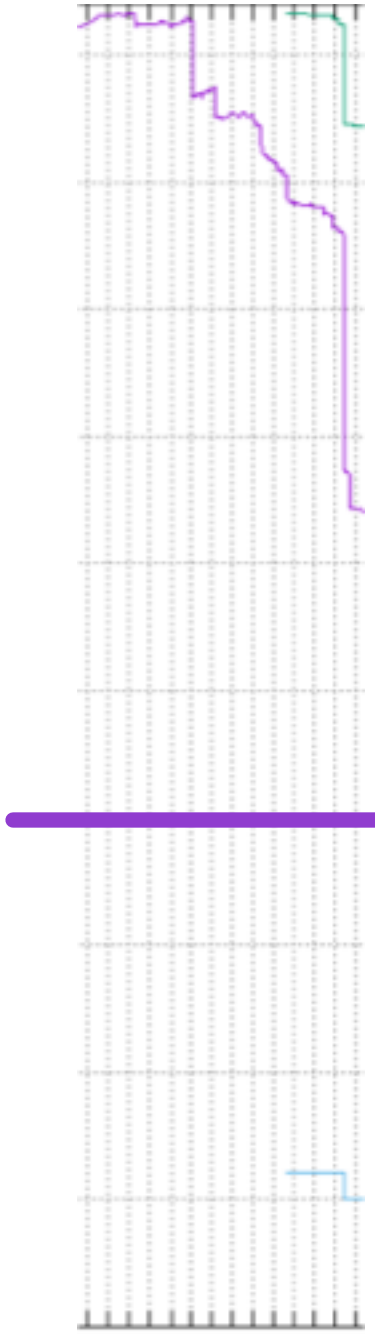
Requirement

- Measure our most important problem: complexity

Rationale

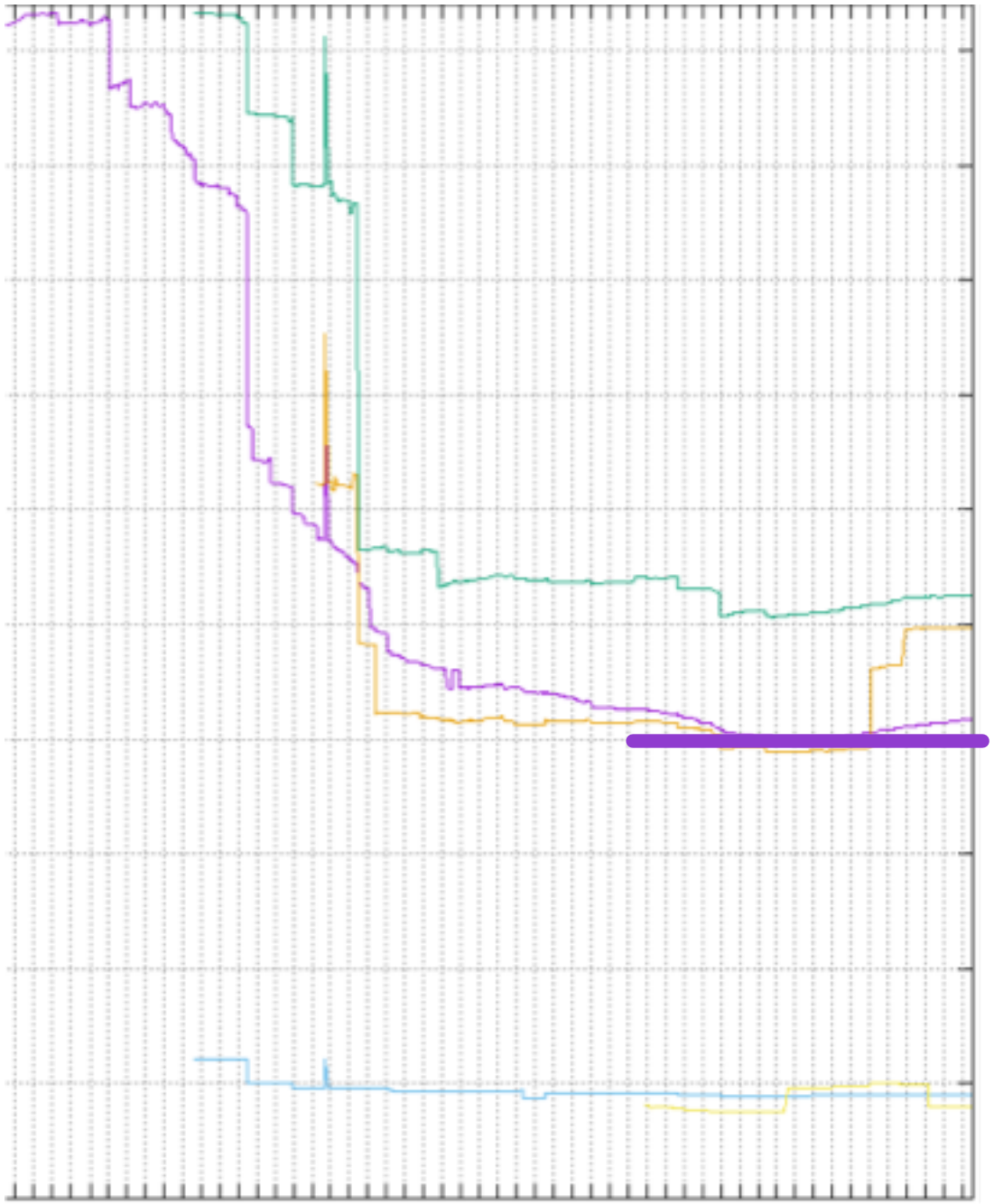
- Binary is huge anyway
  - Larger than I-cache, TLB, predictors, etc.
- We measured
  - Throughput delta was in the noise
  - Micro benchmarks of course show changes

Kobayashi Maru!

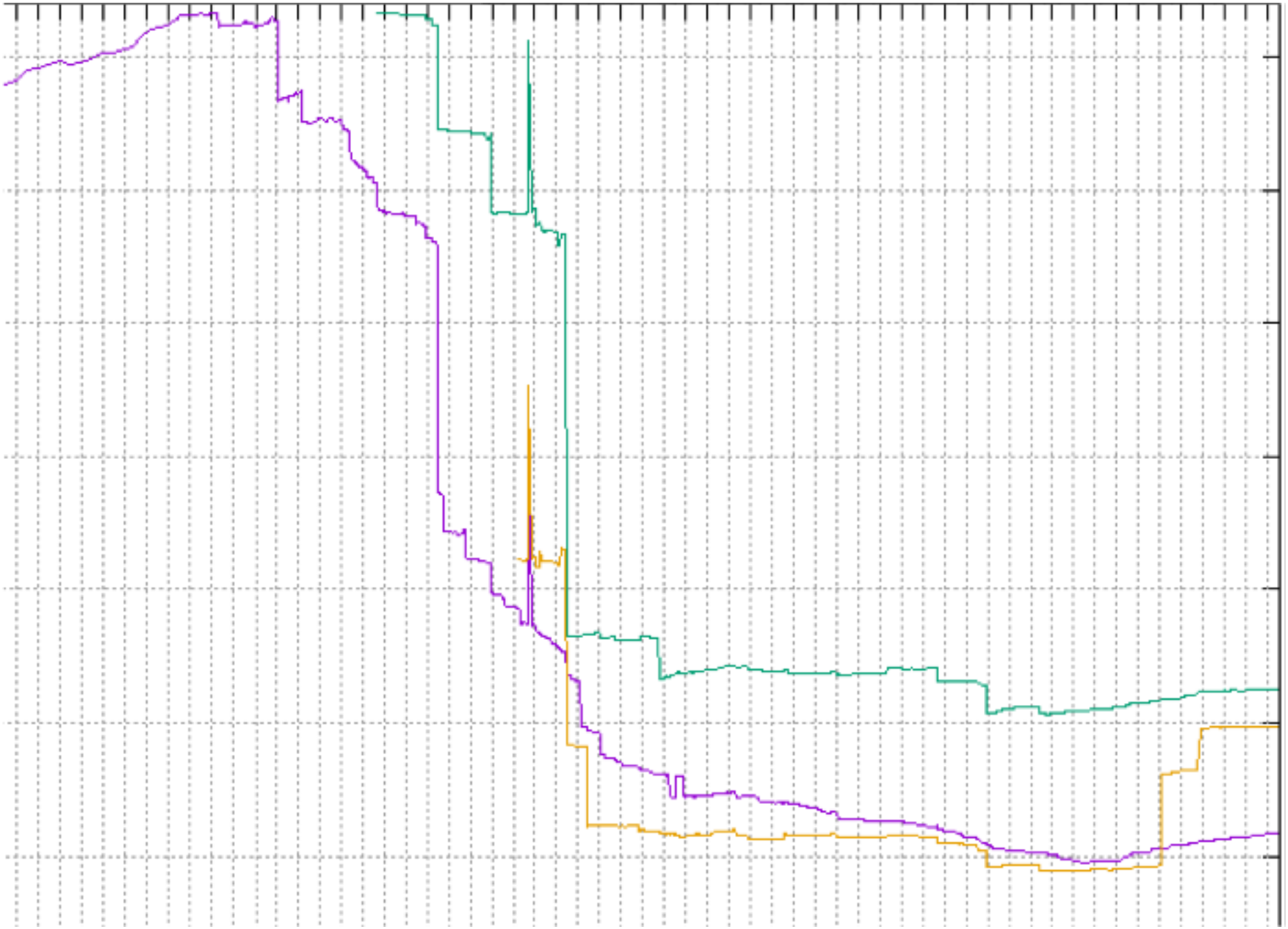


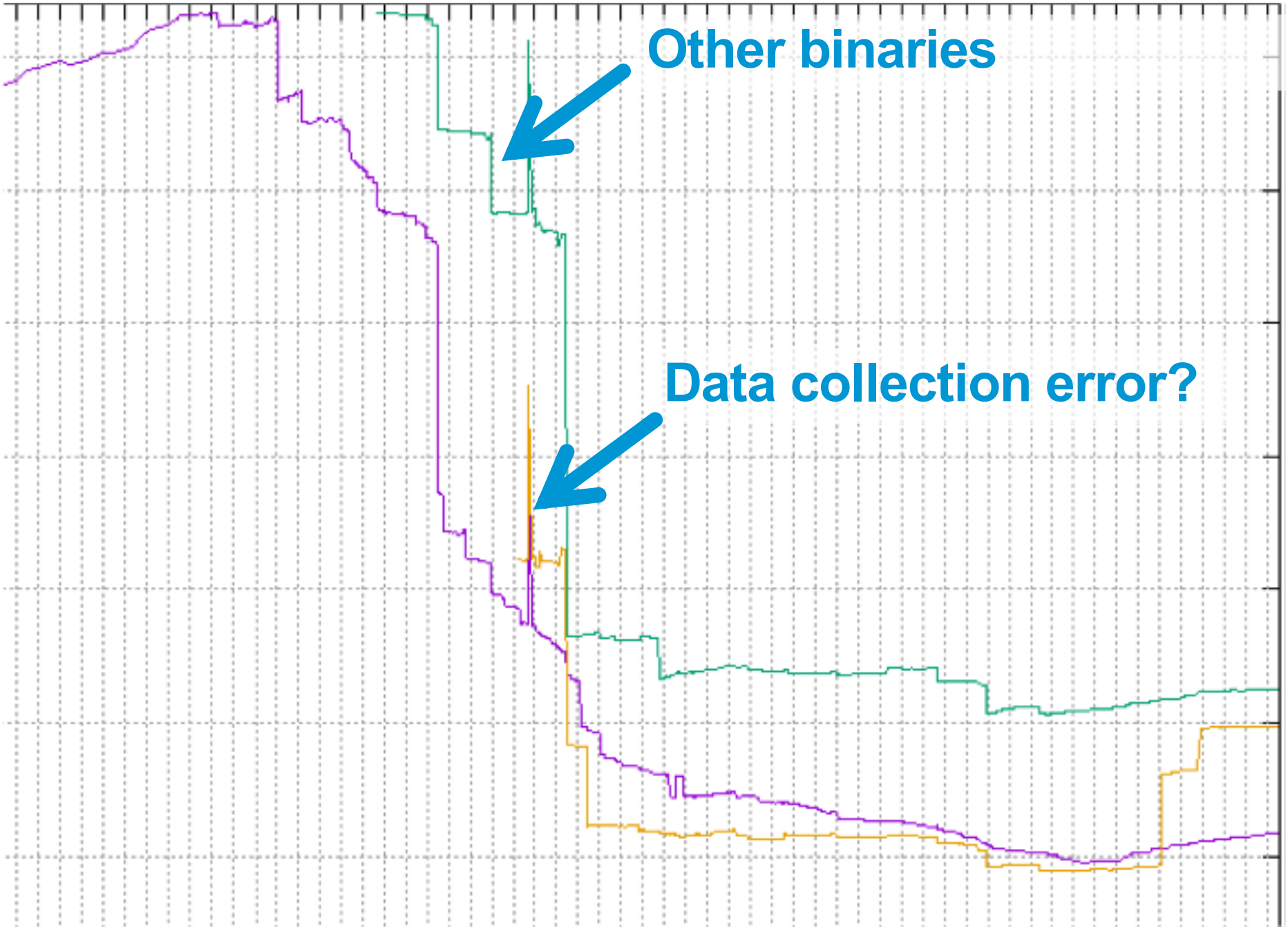
40% - goal

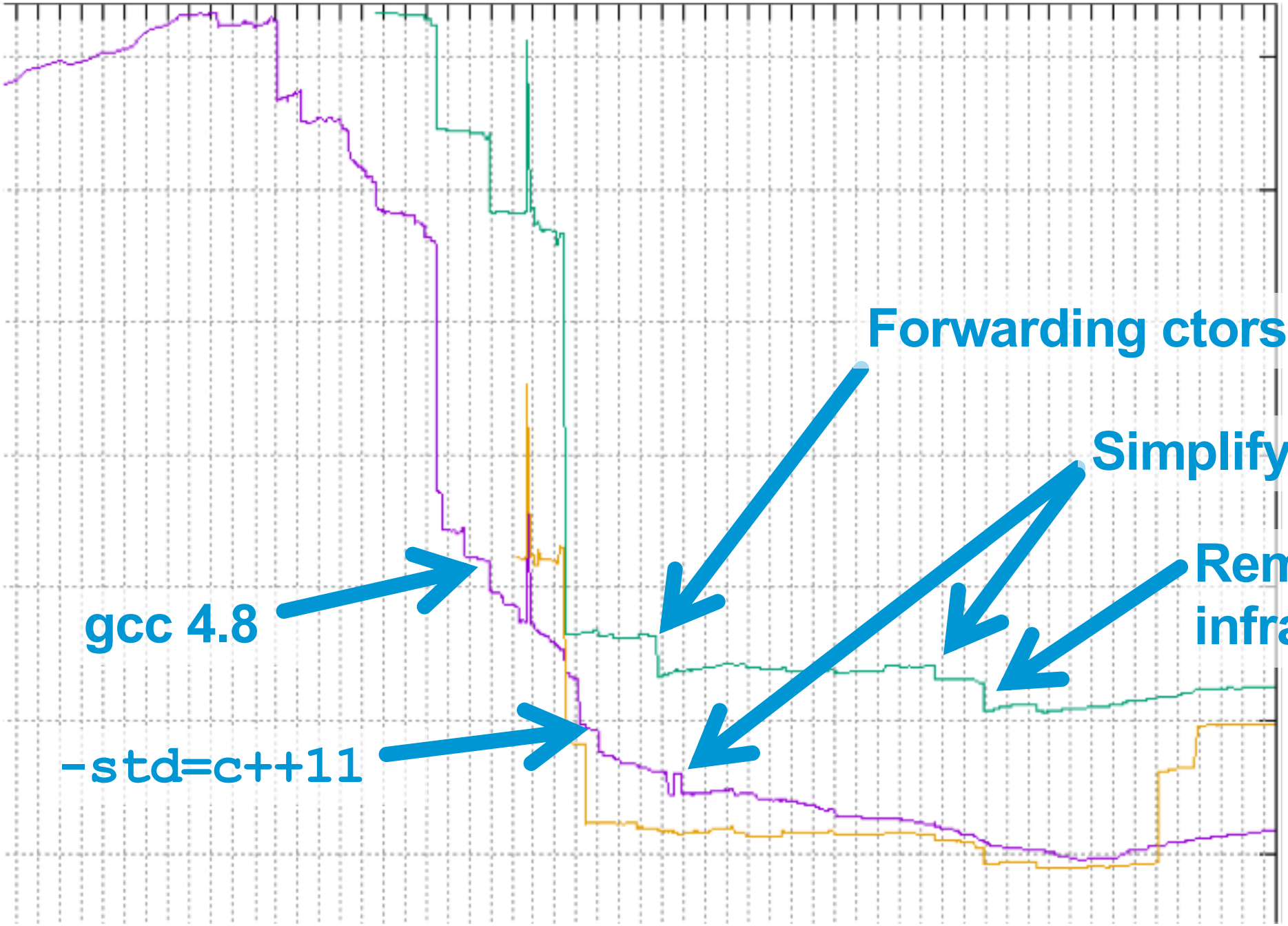
How does this movie end?

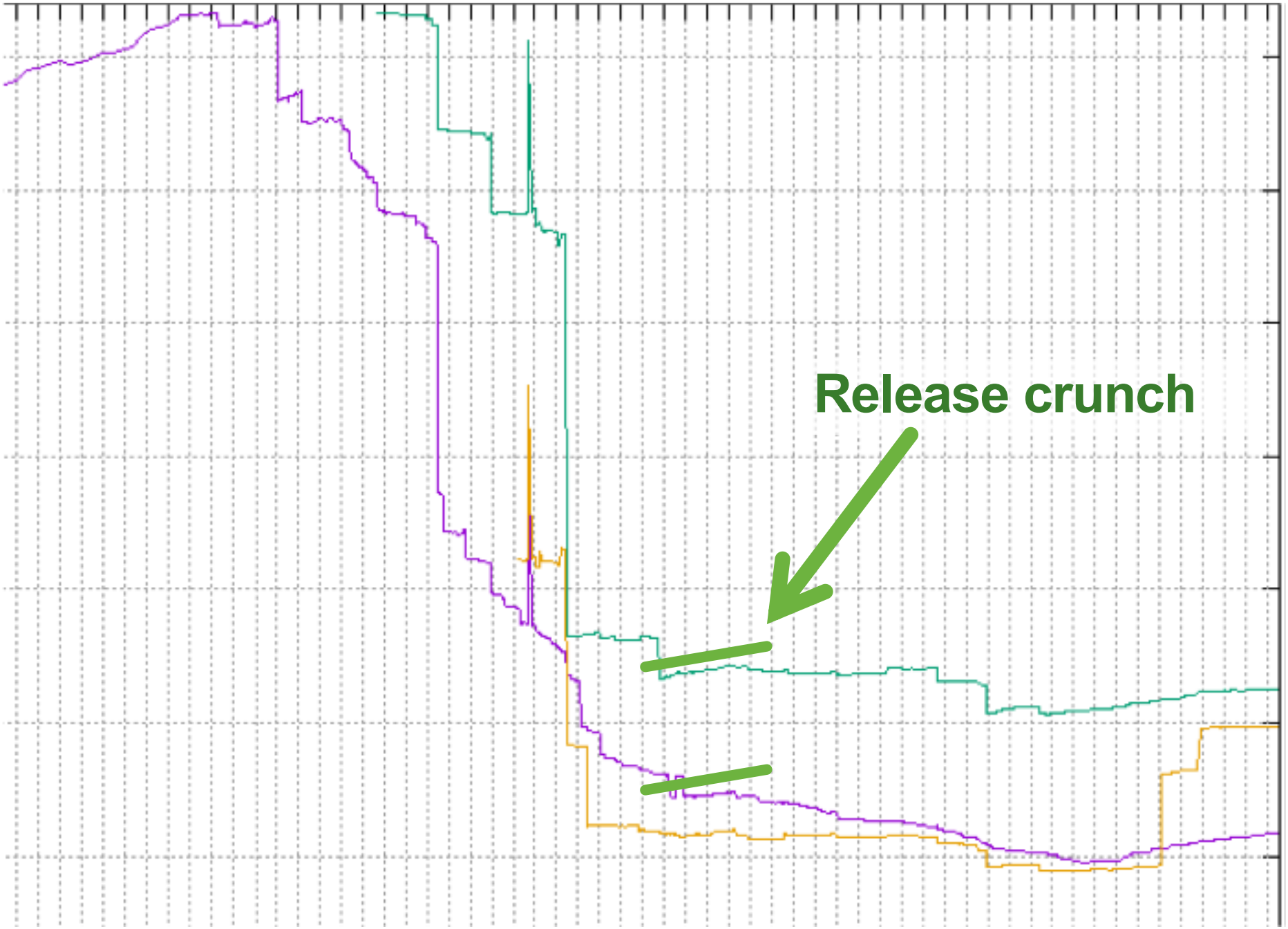


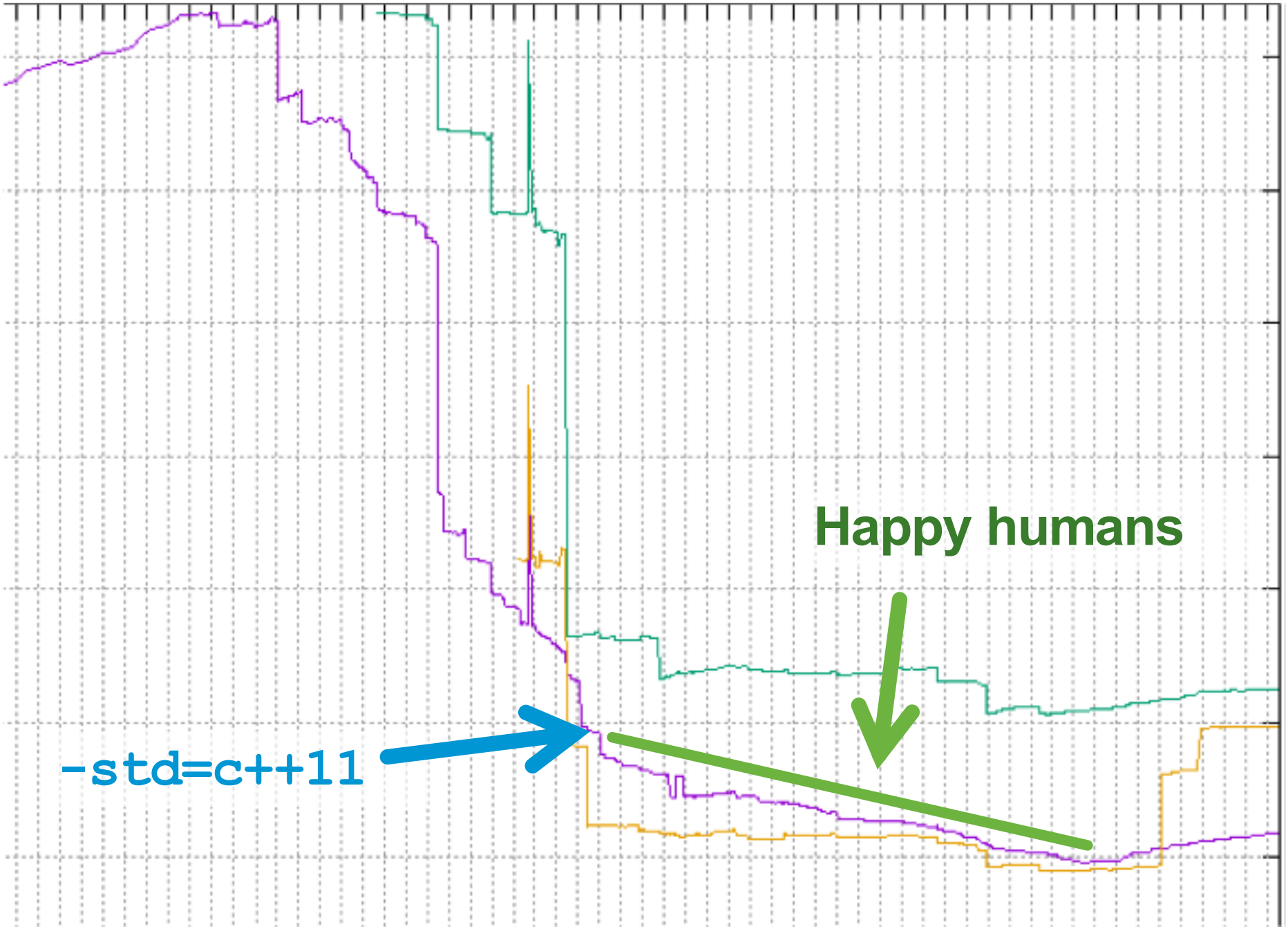
40% - goal




Other binaries
Data collection error?


Forwarding ctors
Simplify class hierarchy
Remove unused infrastructure
gcc 4.8
-std=c++11


Release crunch


Happy humans
-std=c++11

Real Work Too

- Algorithmic optimization
- Removing whole subsystems
- Adding entirely new features

C++11 Things

- out parameters -> return values
- Reference types to value types
- `bind` -> lambda
- Range based for
- `final`

General Things

- Simplify class structure
- Remove dead features
- Remove layers of indirection
- Remove un-used instrumentation
- Remove simulators and test code
- Rationalize logging
- Exception factories
- Etc.

```
app: Add frob::ping

* Add the ping method to the frob service.

Testing Done:
* Unit tests
* mylib size:
      text       data        bss        dec   filename
    196527      21900      74873     293300   mylib-after
    194647      21900      74873     291420   mylib-before
      2120          0          0       2120 bytes
```

```
app: Add frob::pong

* Add the pong method to the frob service.

Testing Done:
* Unit tests
* mylib size:
      text       data        bss        dec   fil
    206527      21900      74873     303300   mylib-after
    196647      21900      74873     293420   mylib-before
     10120          0          0      10120 bytes
```

Review

- Small object copies
- Class design
- Use a library
- Avoid library
- Etc.

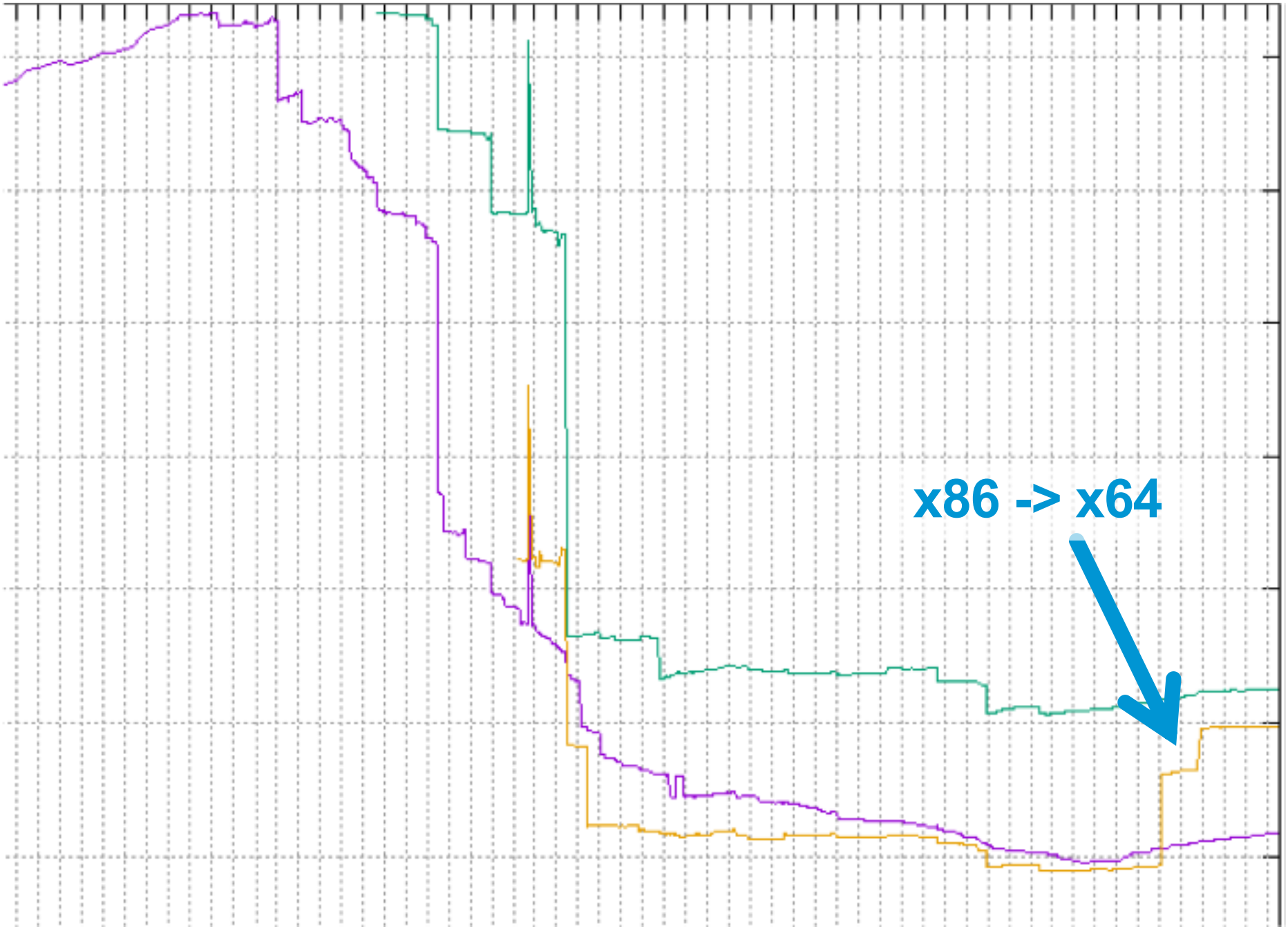
Self Review

- If there are two ways, choose the smaller one.

The Rule is to Measure

The Measure is Not the Rule

Sometimes an increase in size is a decrease in complexity


x86 -> x64

The Measure is not the Rule

- `bind` -> lambda
- RAII
- Range based for
- `unique_ptr`
- `<algorithm>`
- First instantiation of a template
- Etc.

Outside of VMware

cppcon | 2016

THE C++ CONFERENCE • BELLEVUE, WASHINGTON

Rich Code For Tiny Computers:

A Simple Commodore 64 Game in C++ 17

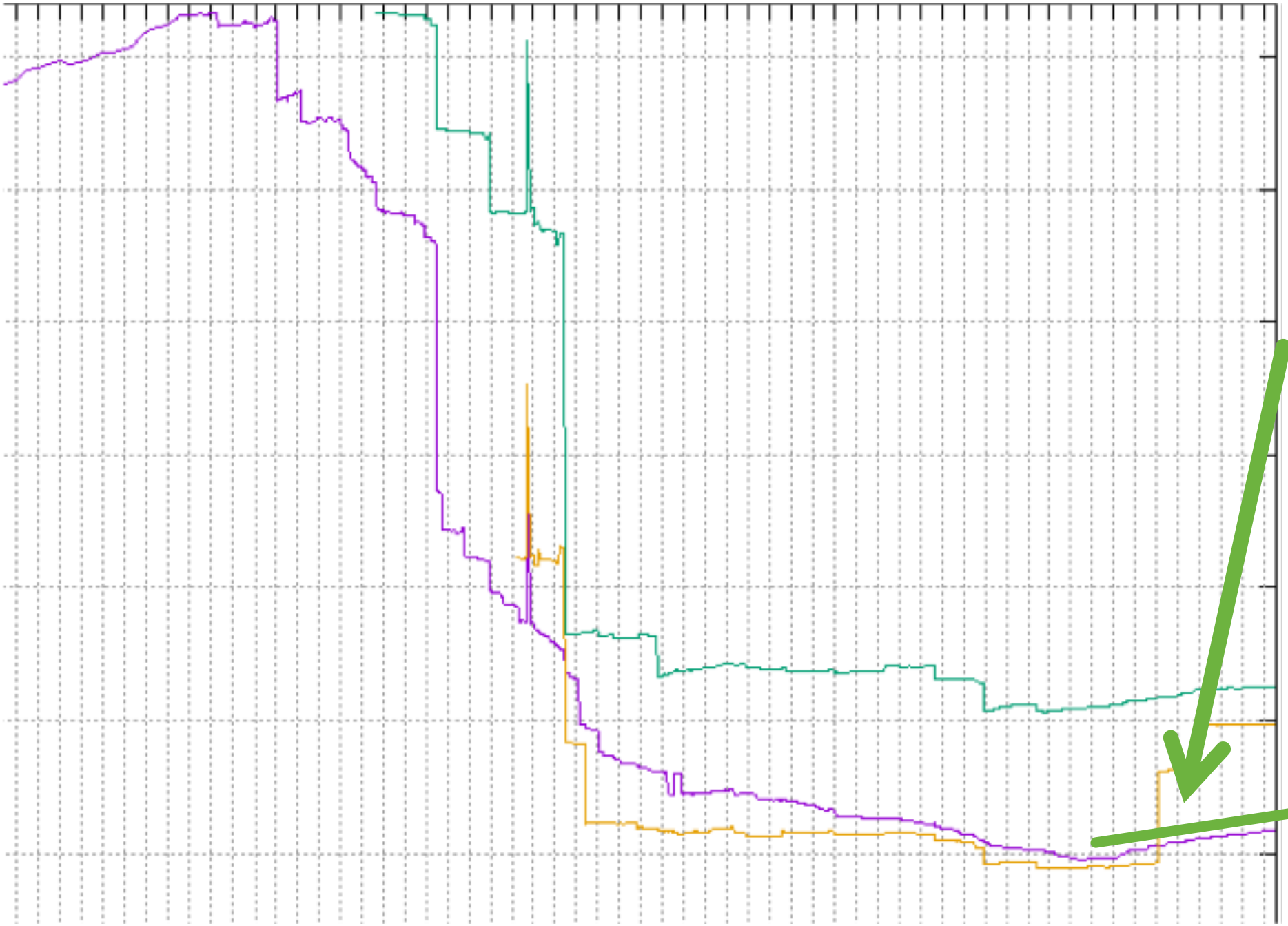
```
78
79          VIC_II &vic;
80        };
81
82        return Frame(*this);
83      }
84    };
85
86  }
87
88  int main()
89  {
90    VIC_II vic;
91
92    while (true) {
93      const auto frame = vic.frame();
94      if (const auto joy = JoyStick(1); joy.fire)
95      {
96        ++vic.background();
97      }
98    }
99  }
100
101
102
103
104
```

```
1  main:
   # @main
2          jmp     .LBB0_1
3  .LBB0_4:
   # %if.end
4          incb    53280
5  .LBB0_1:
   # %while.cond.i.i
6          movzbl  53281, %eax
7          cmpb    $-16, %al
8          jne     .LBB0_1
9          testb   $16, 56321
10         jne     .LBB0_4
11         incb    53281
12         jmp     .LBB0_4
13
14
15
```

Outside of VMware

https://github.com/google/bloaty

Wrapping Up



- Business cycle
- Adding value

Future Directions

- Bloaty!
  - We have scripts of a certain age
  - `size` has an arbitrary notion of .text
  - Select and disassemble
  - Bloaty should be in your toolbelt.

Future Directions

- Back to `-O2`?
  - Occasional Gotchas
  - Some builds really should be `-Os`
  - Test what you deploy
  - Combine with clang plugins
  - Build system integration

Summing Up

- Managing complexity is part of the product lifecycle
- `-Os` size is a convenient proxy for complexity
- Make it part of the edit, test loop
- Make it part of the review cycle
- Make it part of design

Questions?

Mark Zeren
mzeren@vmware.com
cpplang.slack.com

vmware