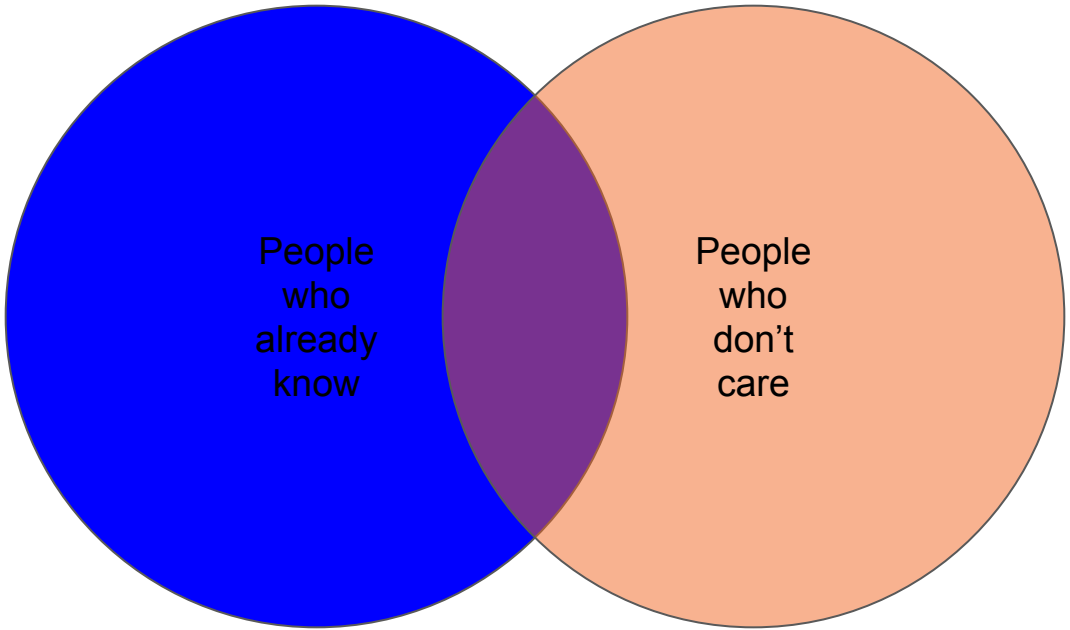
Optimizing Away Virtual Functions May Be Pointless

SHACHAR SHEMESH

Slide #2 - who am I?




DO NOT
ENTER

Simple benchmark

```
class Base {
public:
   int concrete() const;
   virtual int virt() const;
};
```

```
void benchmark(Base *b, size_t num_iterations)
{
   auto start = Clock::now();
   for( size_t i=0; i<num_iterations; ++i ) {
       b->concrete();
   }
   auto end = Clock::now();

   auto concrete_duration = end-start;

   start = Clock::now();
   for( size_t i=0; i<num_iterations; ++i ) {
       b->virt();
   }
   end = Clock::now();

   auto virt_duration = end-start;
```

One reviewer’s notes:

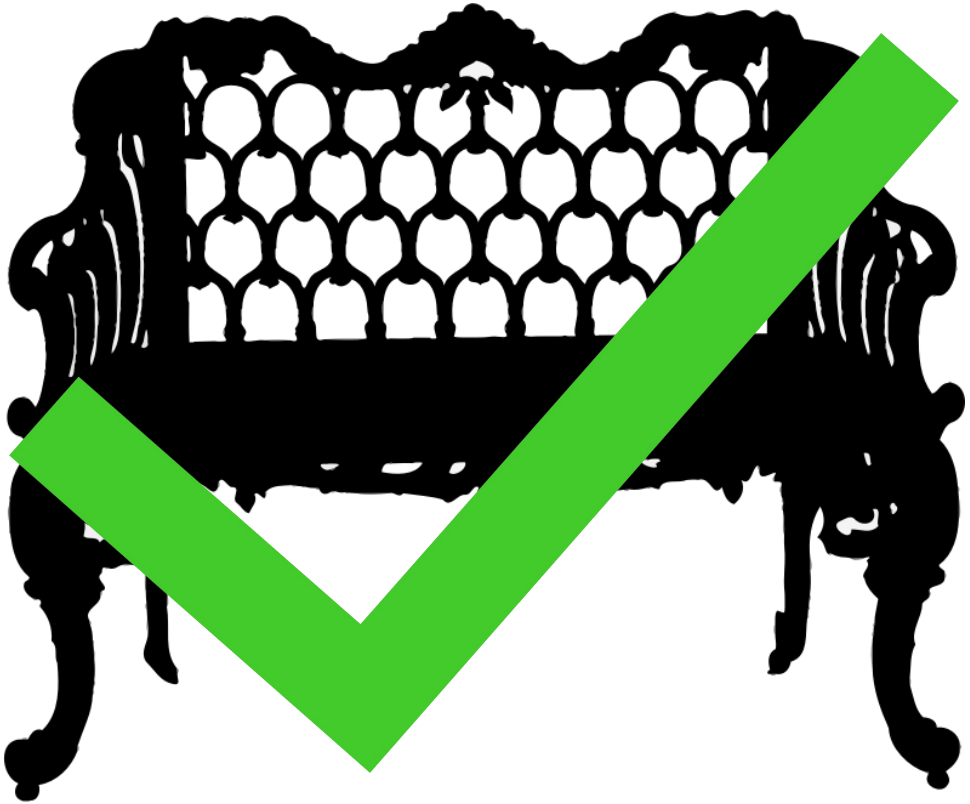
There are interesting technical details and surprising conclusions that virtual functions can be actually faster. Since CPU architectures are mentioned, I'd expect to see deep assembly profiling.

Ok, *some* assembly *is* required

```
for( size_t i=0; i<num_iterations; ++i ) {
12de:       4d 8d 64 24 01          lea    0x1(%r12),%r12
    result ^= b->concrete();
12e3:       e8 a8 ff ff ff          call   1290 <Base::virt() const>
12e8:       41 31 c6                xor    %eax,%r14d
```

```
for( size_t i=0; i<num_iterations; ++i ) {
1353:       0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
    result ^= b->virt();
1358:       48 8b 03                mov    (%rbx),%rax
135b:       48 89 df                mov    %rbx,%rdi
135e:       ff 10                   call   *(%rax)
1360:       31 c5                   xor    %eax,%ebp
```

Benchmark - Two file formats



But I have another computer…

```
shachar@shachar:~/sources/Lectures/cached-benchmark$ ./benchmark2.gcc  vmlinuz
Virtual run took 5893147 resulting in 0, concrete run took 7182348 resulting in 0. Virtual ran -17.9496% slower
shachar@shachar:~/sources/Lectures/cached-benchmark$ ./benchmark2.clang  vmlinuz
Virtual run took 3801467 resulting in 0, concrete run took 4129302 resulting in 0. Virtual ran -7.93924% slower
```

Different CPUs

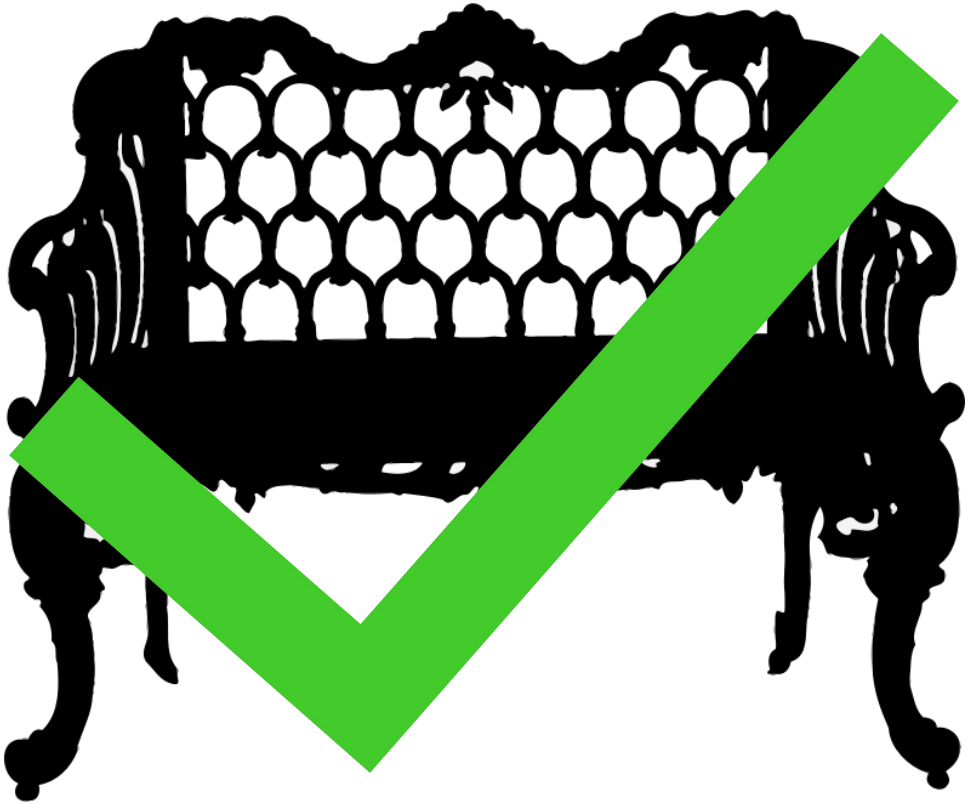
Laptop:

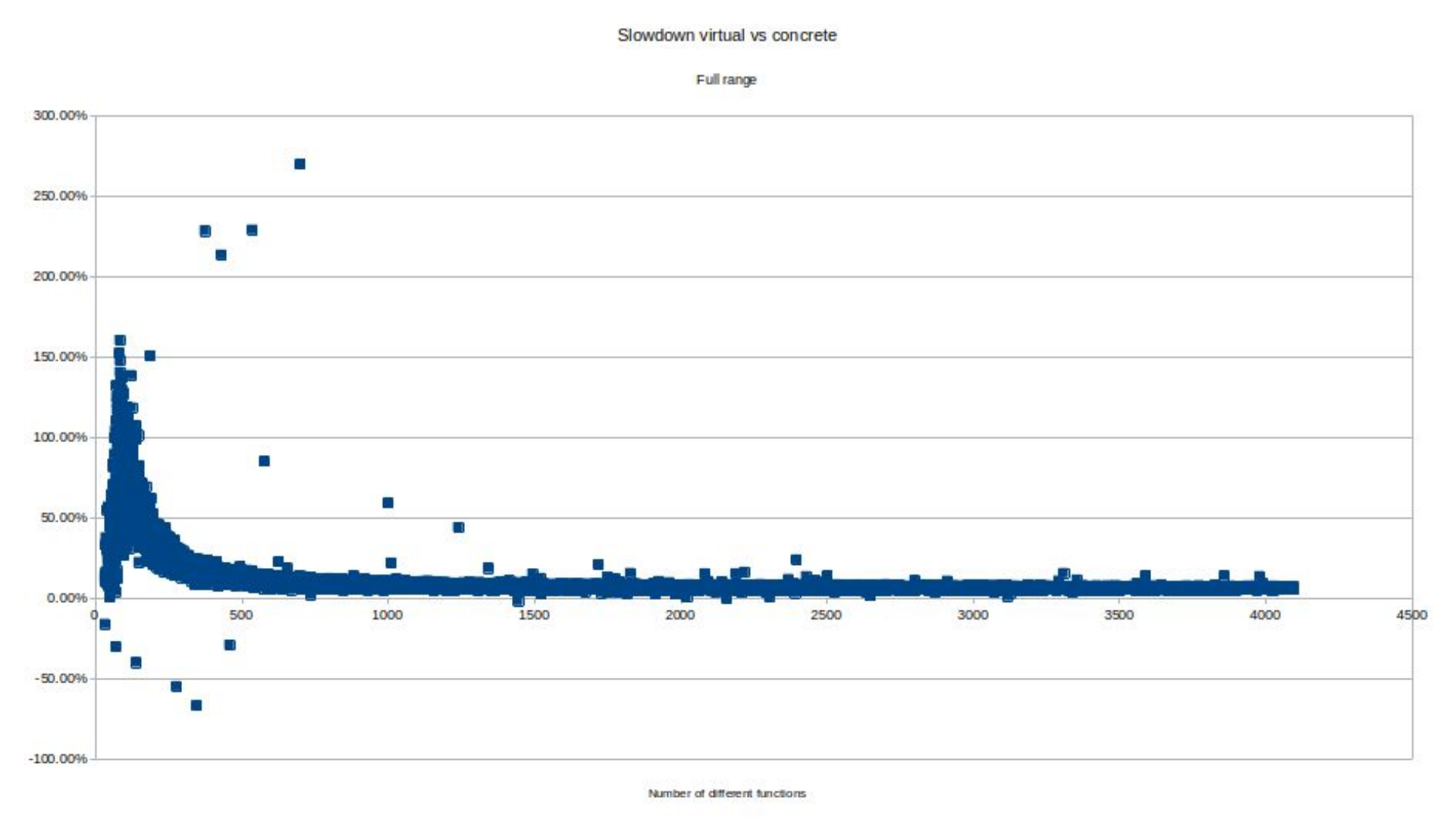
```
Model name:            Intel(R) Core(TM) i5-10310U CPU @ 1.70GHz
  Thread(s) per core:  2
  Core(s) per socket:  4
  Stepping:            12
```

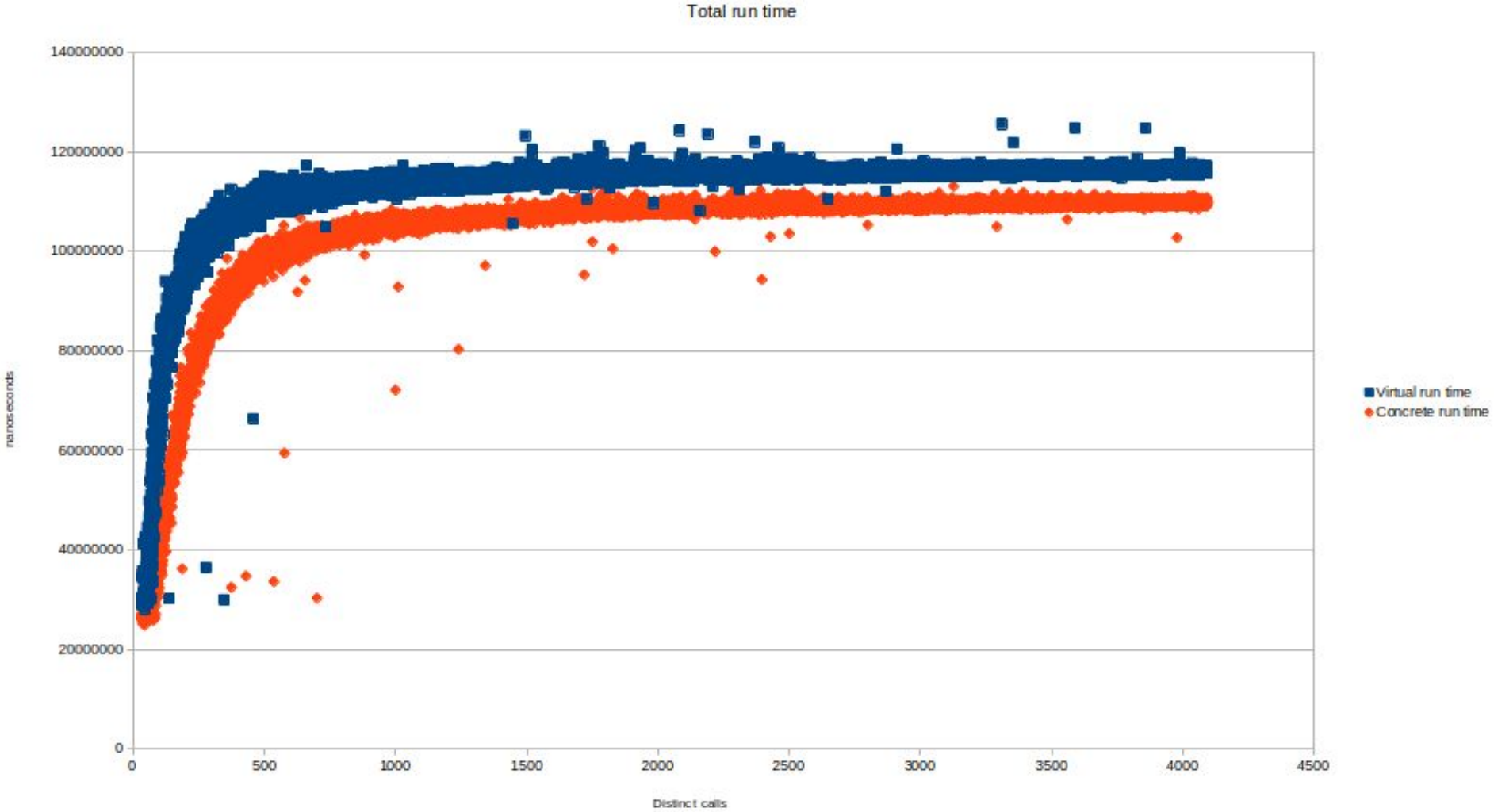
Desktop:

```
Model name:            AMD Ryzen 9 5900X 12-Core Processor
  Thread(s) per core:  2
  Core(s) per socket:  12
  Stepping:            0
```

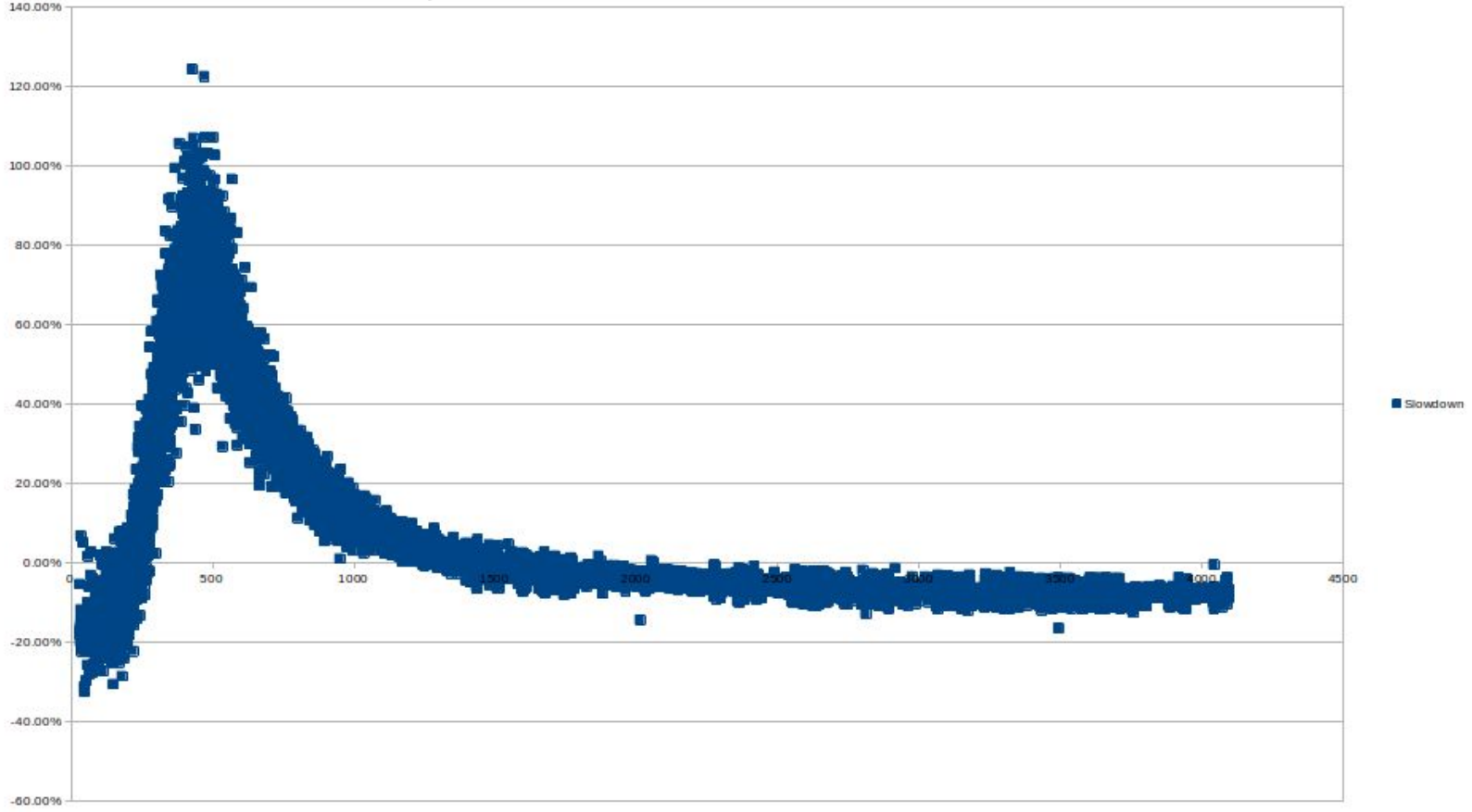
Benchmark - Taking Things to Extreme



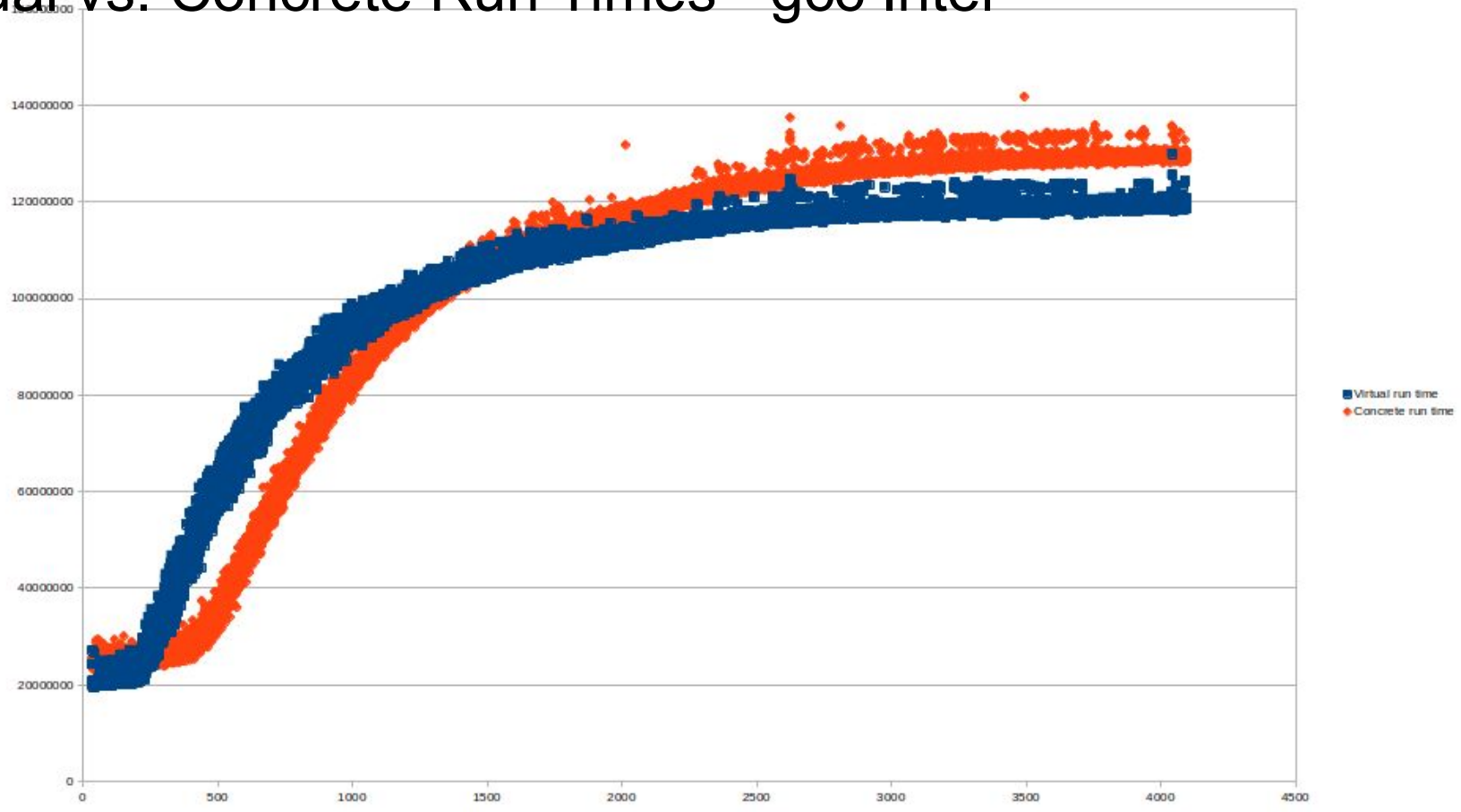

Slowdown virtual vs concrete
Full range
300.00%
250.00%
200.00%
150.00%
100.00%
50.00%
0.00%
-50.00%
-100.00%
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Number of different functions


Total run time
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
nanoseconds
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Distinct calls
Virtual run time
Concrete run time

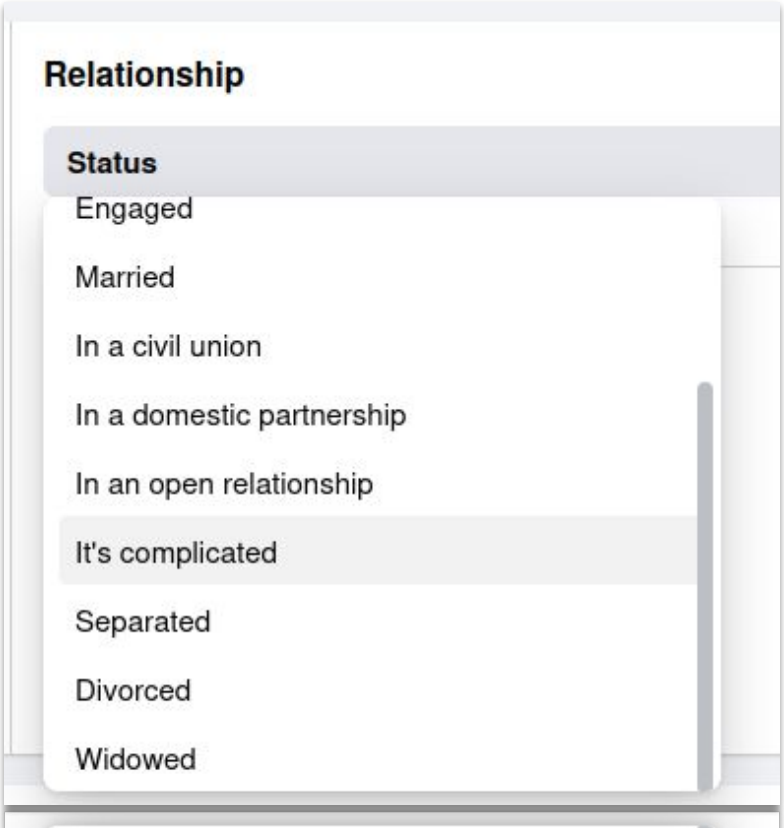
Virtual slowdown - gcc Intel



Virtual vs. Concrete Run Times - gcc Intel



Conclusions



Relevant factors

- CPU manufacturer
- CPU version
- Precise code path
- Temperature(?)
- OS interrupts(?)
- Compiler optimization level
- Compiler flags
- Compiler version
- Compiler

I’m not alone!

https://stackoverflow.com/questions/57726401/stdvariant-vs-inheritance-vs-other-ways-performance

---

I made some changes and the results are very different from compiler to compiler now. But it seems like `std::get_if` and `std::holds_alternatives` are the best solutions. `virtual` seems to work best for unknown reasons with clang now. That really surprises me there because I remember `virtual` being better in gcc. And also `std::visit` is totally out of competition; in this last benchmark it's even worse than vtable lookup.

Does it even matter?

WEKA

DriveU.auto

Conclusions

- The notion that “virtual functions are slower” is flat out wrong.
  - Which is not to say they are faster
  - Some of the suggested alternatives *are* consistently slower
- Don’t benchmark, profile.
- Use the best design for your code. Only reconsider if it’s not fast enough.

More of the same


https://github.com/Shachar/cached-benchmark


https://youtube.com/CompuSAR


https://twitter.com/ShacharShemesh


https://fosstodon.org/@compusar


shachar@shemesh.biz

Also, this talk’s discord channel