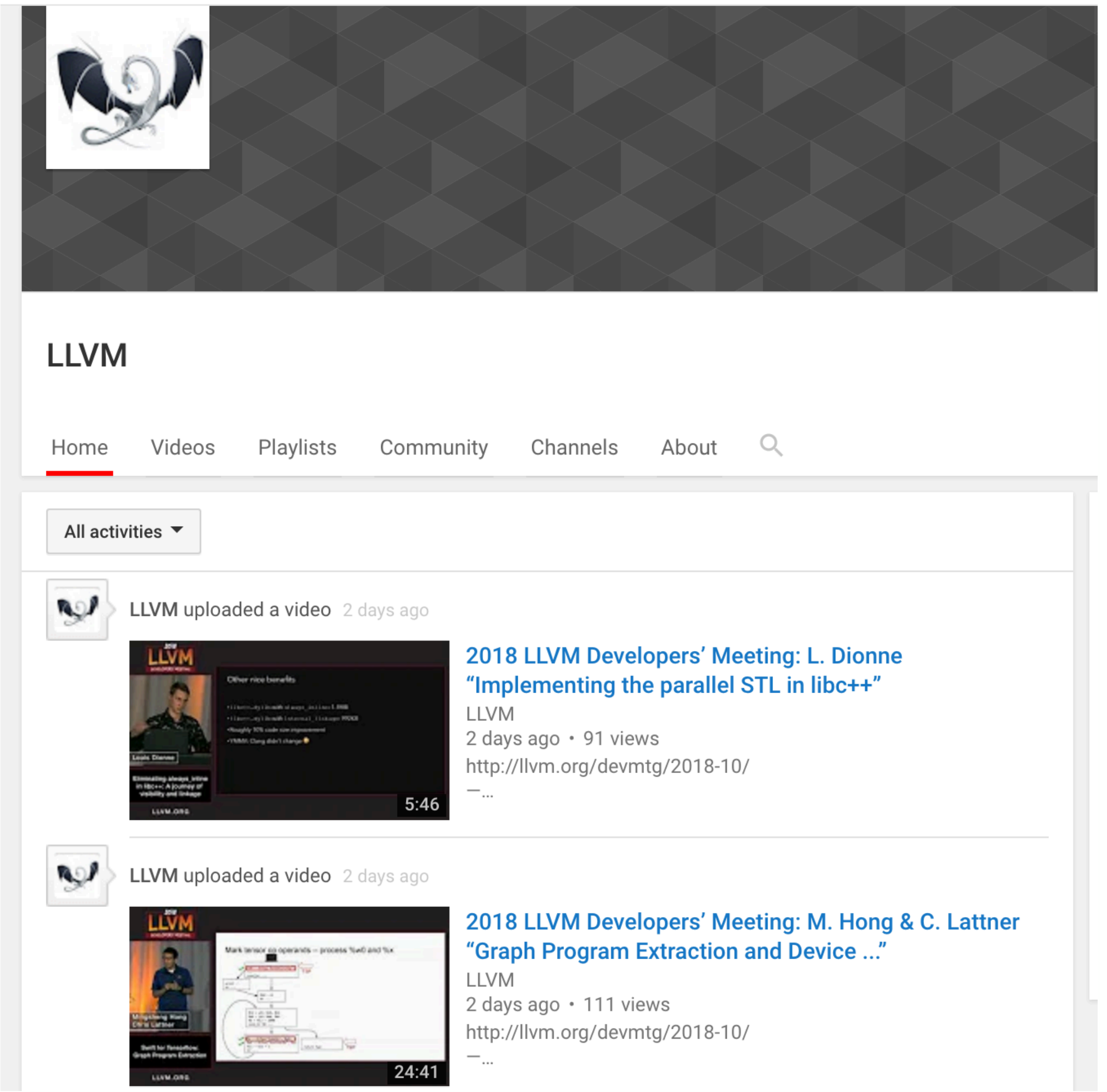
Takeaway from LLVM dev meeting

Yuka Takahashi - Princeton University, CERN

LLVM developers’ meeting

- Annual 2 day meeting for developers in LLVM/Clang and related projects
- Held in San Jose Convention Center, San Jose, CA. October 17-18, 2018

All talks will be uploaded to LLVM Youtube channel ->



LLVM uploaded a video 2 days ago

2018 LLVM Developers’ Meeting: L. Dionne “Implementing the parallel STL in libc++”
LLVM
2 days ago • 91 views
http://llvm.org/devmtg/2018-10/
—...

5:46

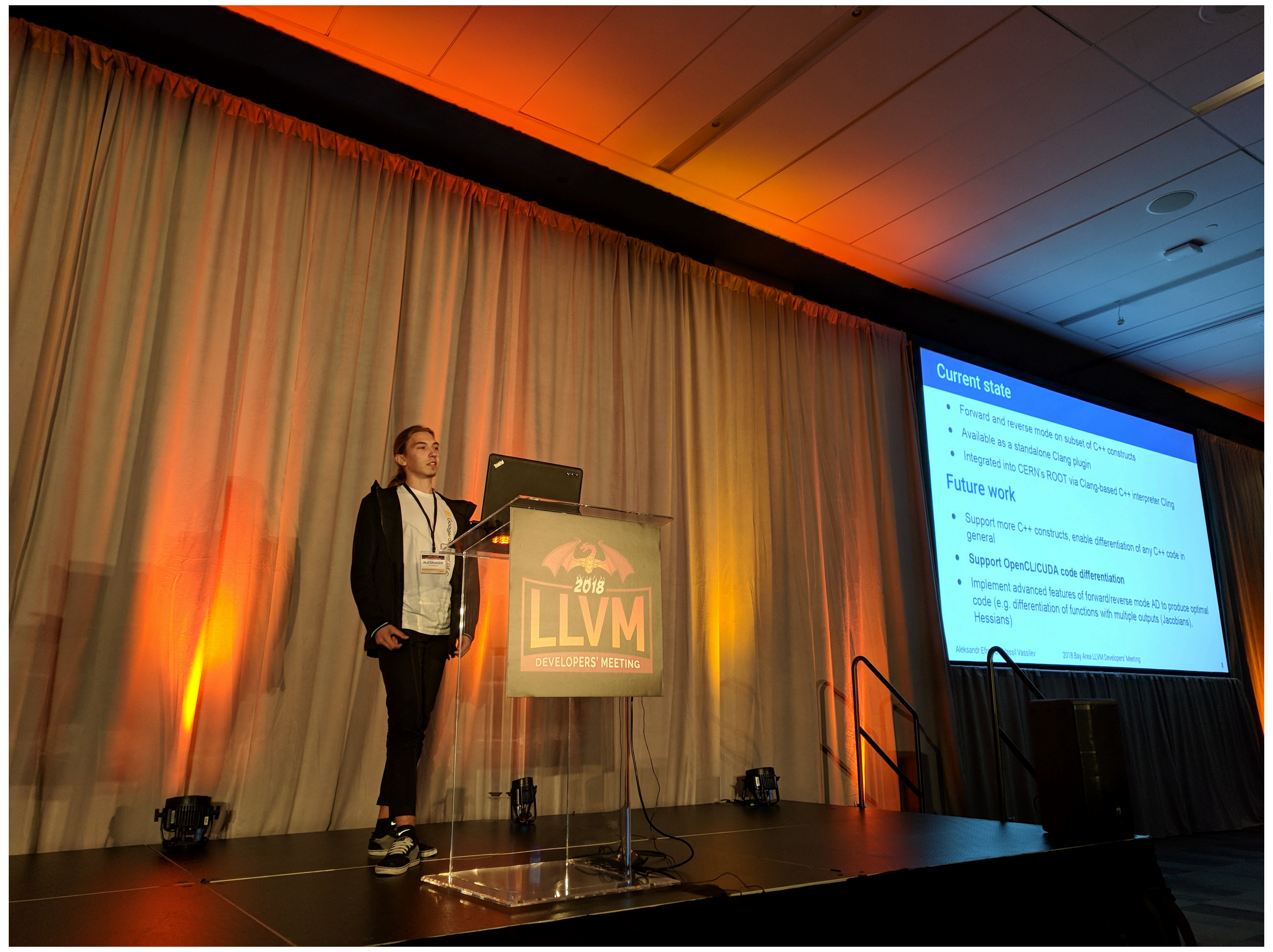
LLVM uploaded a video 2 days ago

2018 LLVM Developers’ Meeting: M. Hong & C. Lattner “Graph Program Extraction and Device ...”
LLVM
2 days ago • 111 views
http://llvm.org/devmtg/2018-10/
—...

24:41

LLVM developers’ meeting

- Went with Vassil and Aleksandr (GSoC student of Vassil on Clad). Vassil and me were funded by Princeton, Aleksandr was funded by LLVM student grant.
- Aleksandr had a Lightning talk about Clad



Current state

- Forward and reverse mode on subset of C++ constructs
- Available as a standalone Clang plugin
- Integrated into CERN's ROOT via Clang-based C++ interpreter Cling

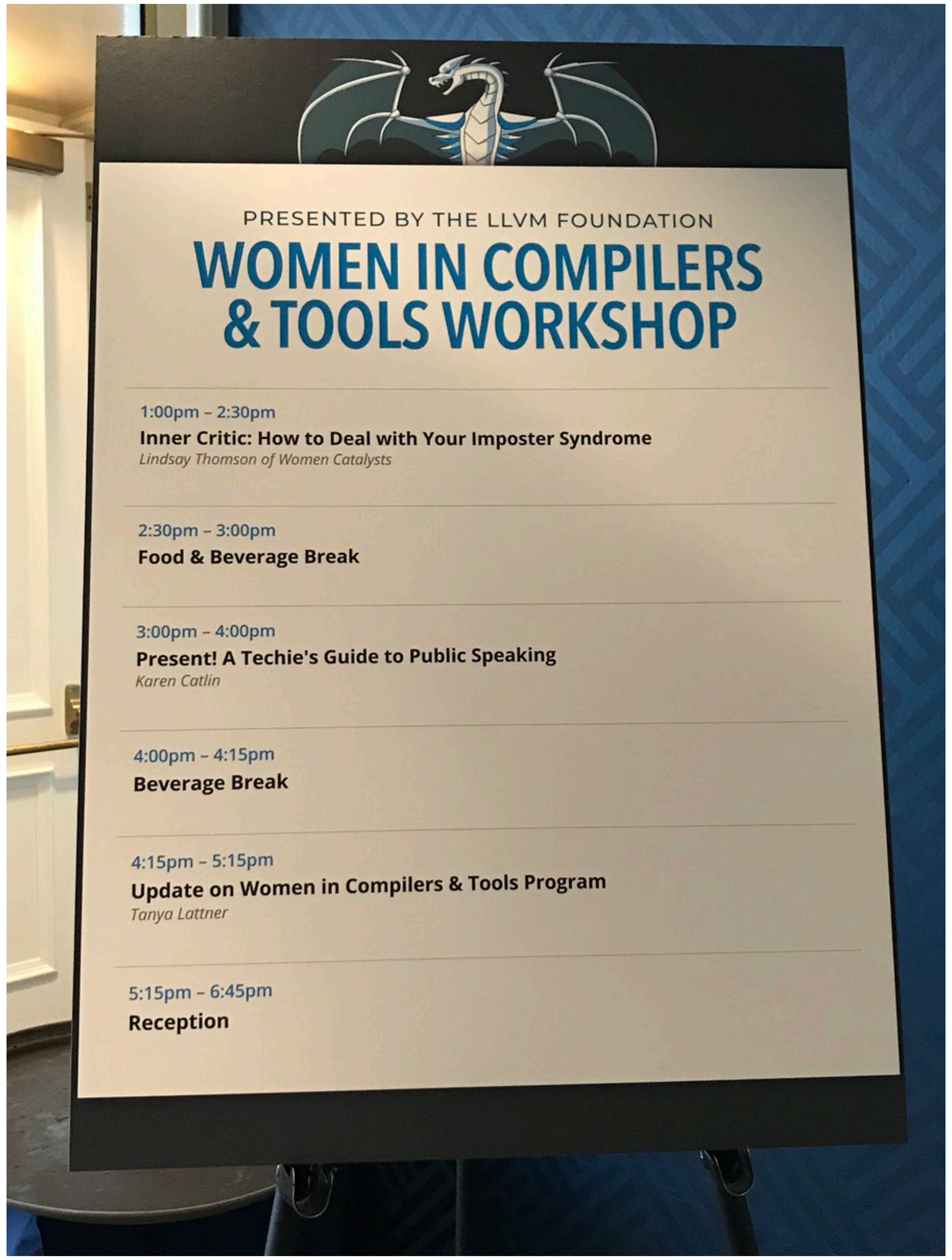
Future work

- Support more C++ constructs, enable differentiation of any C++ code in general
- **Support OpenCL/CUDA code differentiation**
- Implement advanced features of forward/reverse mode AD to produce optimal code (e.g. differentiation of functions with multiple outputs (Jacobians), Hessians)

Aleksandr Efre[illegible] Vassil Vassilev 2018 Bay Area LLVM Developers' Meeting 8

Women in Compilers and Tools workshop

- Held the day before the meeting
- Started last year (It was just a dinner last year, but this year it's a workshop)
- Talk about imposter syndrome and public speaking
- Discussions about how to engage more women to compiler industry



PRESENTED BY THE LLVM FOUNDATION

WOMEN IN COMPILERS & TOOLS WORKSHOP

1:00pm – 2:30pm
Inner Critic: How to Deal with Your Imposter Syndrome
Lindsay Thomson of Women Catalysts

2:30pm – 3:00pm
Food & Beverage Break

3:00pm – 4:00pm
Present! A Techie's Guide to Public Speaking
Karen Catlin

4:00pm – 4:15pm
Beverage Break

4:15pm – 5:15pm
Update on Women in Compilers & Tools Program
Tanya Lattner

5:15pm – 6:45pm
Reception

List of the talks I was interested

1. Sound Devirtualization in LLVM
2. Heap to Stack conversion
3. VecClone Pass: Function Vectorization via LoopVectorizer
4. Build Impact of Explicit and C++ Standard Modules
5. More efficient LLVM devs: 1000x faster build file generation, -j1000 builds, and O(1) test execution
6. Repurposing GCC Regression for LLVM Based Tool Chains
7. Memory Tagging, how it improves C++ memory safety, and what does it mean for compiler
8. Glow: LLVM-based machine learning compiler
9. Graph Program Extraction and Device Partitioning in Swift for TensorFlow
10. Understanding the performance of code using LLVM's Machine Code Analyzer (llvm-mca)
11. Lessons Learned Implementing Common Lisp with LLVM over Six Years
12. Migrating to C++14, and beyond!
13. Developer Toolchain for the Nintendo Switch

List of the talks I was interested

1. **Sound Devirtualization in LLVM**
2. Heap to Stack conversion
3. VecClone Pass: Function Vectorization via LoopVectorizer
4. Build Impact of Explicit and C++ Standard Modules
5. More efficient LLVM devs: 1000x faster build file generation, -j1000 builds, and O(1) test execution
6. **Repurposing GCC Regression for LLVM Based Tool Chains**
7. **Memory Tagging, how it improves C++ memory safety, and what does it mean for compiler**
8. Glow: LLVM-based machine learning compiler
9. Graph Program Extraction and Device Partitioning in Swift for TensorFlow
10. Understanding the performance of code using LLVM's Machine Code Analyzer (llvm-mca)
11. **Lessons Learned Implementing Common Lisp with LLVM over Six Years**
12. Migrating to C++14, and beyond!
13. Developer Toolchain for the Nintendo Switch

Memory Tagging, how it improves C++ memory safety, and what does it mean for compiler
- Kostya Serebryany (Google)

Find video at: https://youtu.be/iP_iHroclgM
Related paper: https://arxiv.org/abs/1802.09517

Kostya likes improving the quality of existing codebase. He has many more interesting talks on Youtube. (asan, ubsan, fuzzing, testing…) Very educational.

Memory Tagging, how it improves C++ memory safety, and what does it mean for compiler
- Kostya Serebryany (Google)

Find video at: https://youtu.be/iP_iHroclgM
Related paper: https://arxiv.org/abs/1802.09517

Obvious goal of this talk: Motivate CPU and OS vendors to support Memory Tagging

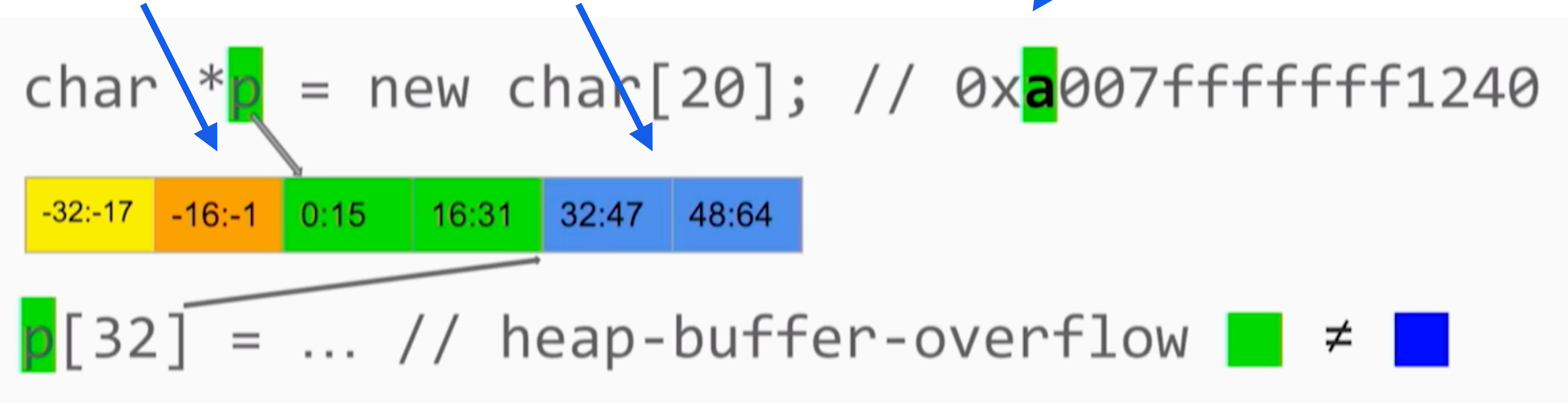
- **50%** of critical security bugs in chrome & android are memory related

Memory Tagging, how it improves C++ memory safety, and what does it mean for compiler
- K. Serebryany (Google)

4 bits tag in pointer upper bits

Previous granule (tagging unit)

Next granule (tagging unit)



```
char *p = new char[20]; // 0xa007fffffff1240
```

-32:-17	-16:-1	0:15	16:31	32:47	48:64

```
p[32] = ... // heap-buffer-overflow 🟩 ≠ 🟦
```

From Kostya’s slide

Memory Tagging, how it improves C++ memory safety, and what does it mean for compiler - K. Serebryany (Google)

- RAM overhead: 3%-5%
- CPU ovrehead: low-single-digit %

- SPARC supported
- ARM has a document

- Need to reduce malloc/free overhead

Relax and wait for the hardware?

No, compiler writers need to reduce the overhead

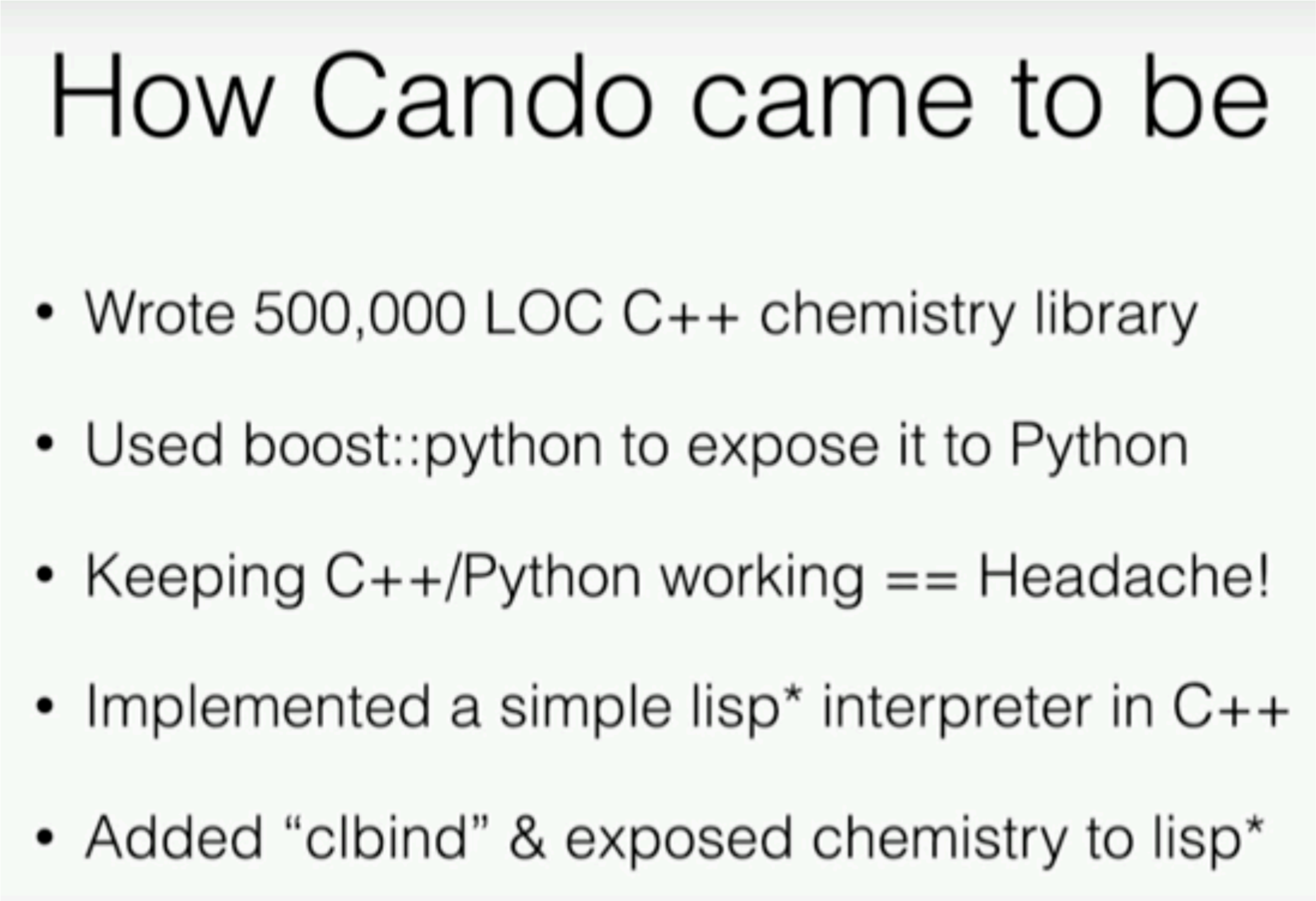
From Kostya's slide

Lessons Learned Implementing Common Lisp with LLVM - C. Schafmeister (Professor at Temple University)

Find video at: https://youtu.be/mbdXeRBbgDM

Christian is a Biological Chemistry researcher
Designing useful molecule by simulation
-> At some point he decided to write **common lisp compiler from scratch** using LLVM backend :D :D

Lessons Learned Implementing Common Lisp with LLVM - C. Schafmeister (Professor at Temple University)



How Cando came to be

- Wrote 500,000 LOC C++ chemistry library
- Used boost::python to expose it to Python
- Keeping C++/Python working == Headache!
- Implemented a simple lisp* interpreter in C++
- Added "clbind" & exposed chemistry to lisp*

From Christian's talk

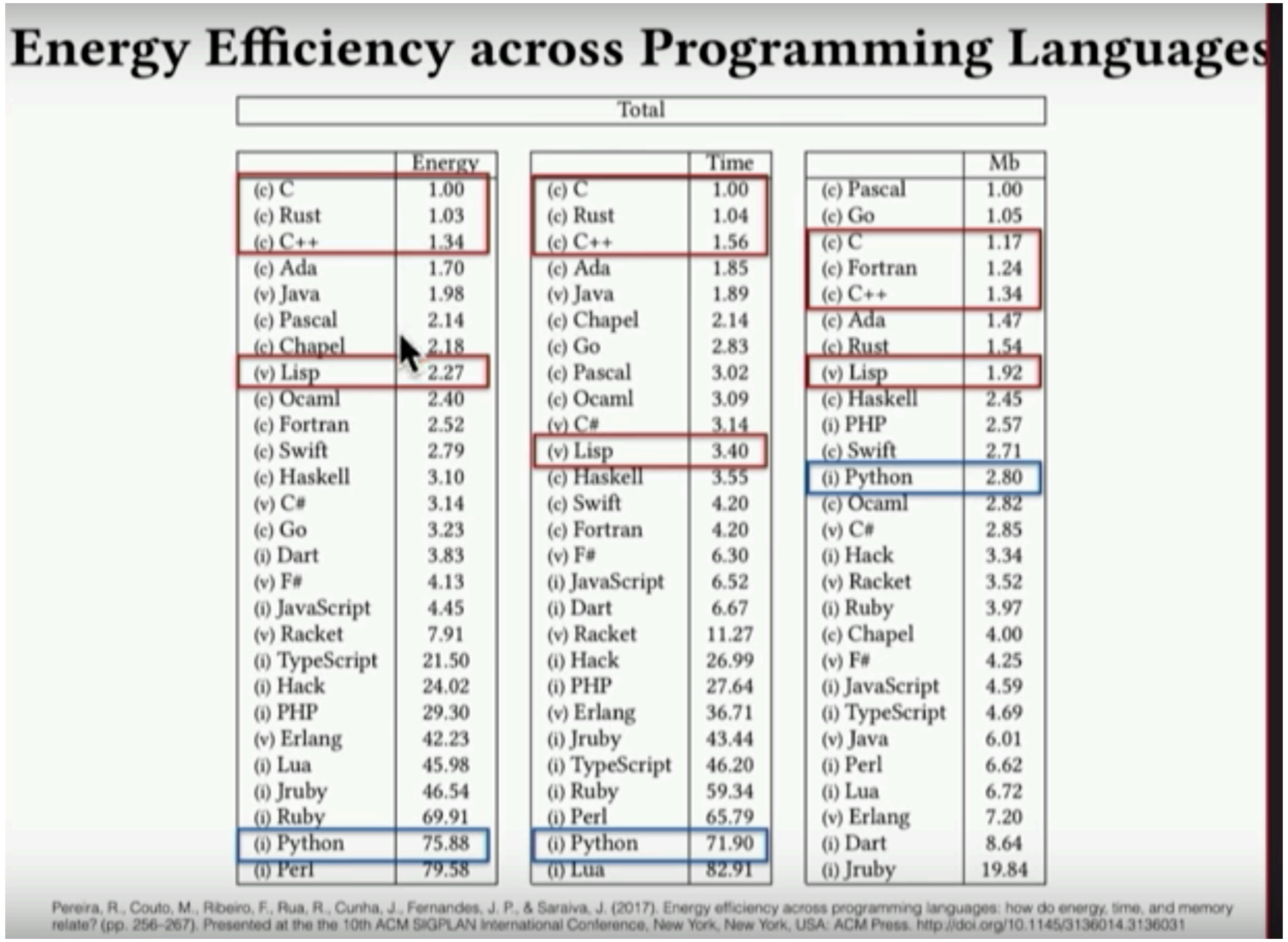
- Got tired with C++/Python binding, he tested lisp with 40,000 CPU supercomputers and it worked amazingly
- Lisp is wonderful with molecules, molecules are graphs and trees, and lisp is good at them
- Wrote C++ lisp binding, so that he can use C++ library from lisp

Lessons Learned Implementing Common Lisp with LLVM - C. Schafmeister (Professor at Temple University)

Why common lisp?
- Macros and compile-time computing

Of all the dynamic languages down there, lisp is only language way up there

Lesson learned: You can develop a compiler based on LLVM **solely from the information on the internet**



Energy Efficiency across Programming Languages

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J. P., & Saraiva, J. (2017). Energy efficiency across programming languages: how do energy, time, and memory relate? (pp. 256–267). Presented at the the 10th ACM SIGPLAN International Conference, New York, New York, USA: ACM Press. http://doi.org/10.1145/3136014.3136031

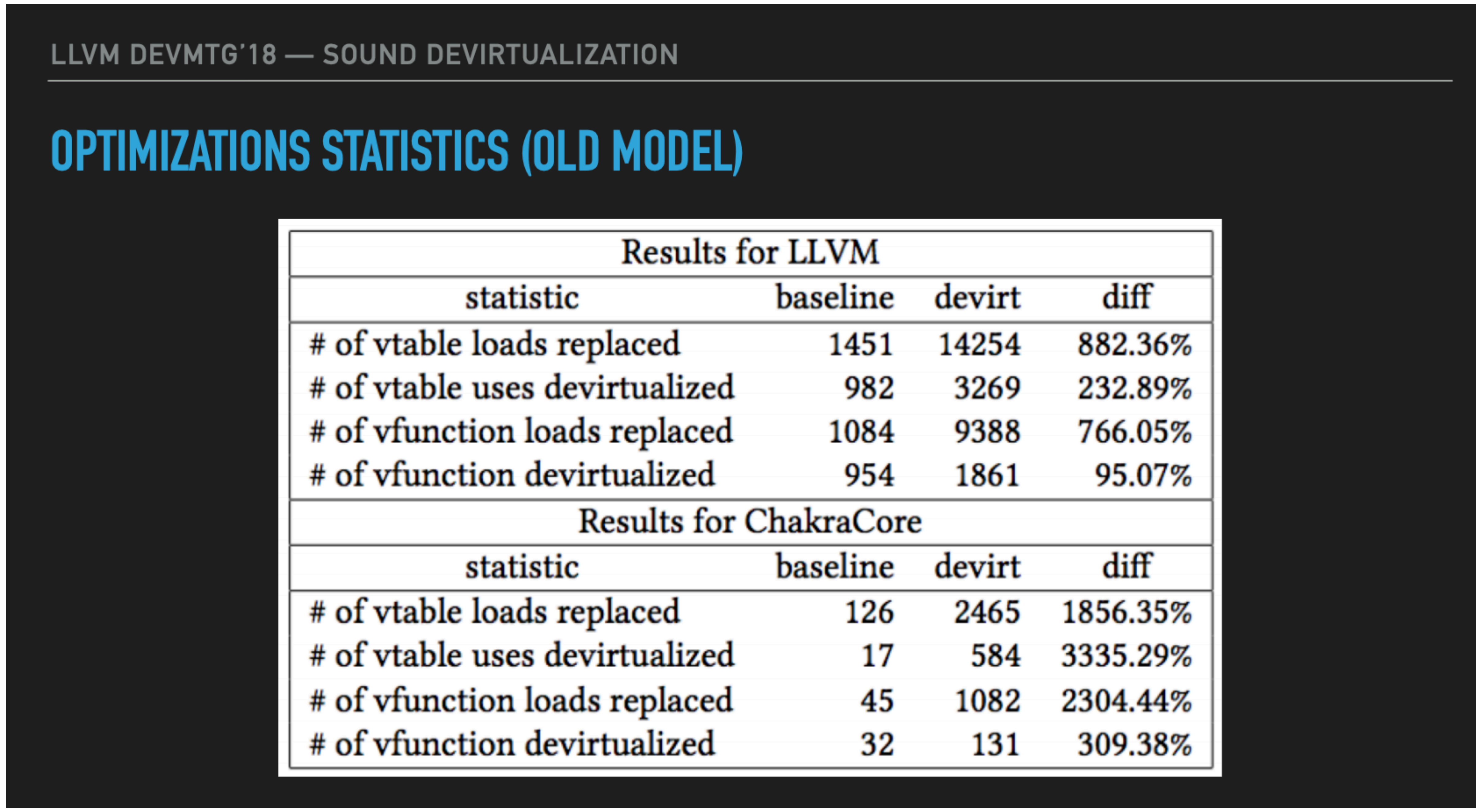
From Christian's talk

Sound Devirtualization in LLVM - Piotr Padlewski

- Devirtualization is an optimization transforming virtual calls into direct calls
  - Clang Devirtualization v1: -fstrict-vtable-pointers
    - Not sound for C++
    - https://dl.acm.org/citation.cfm?id=3135947
  - Clang Devietualizaition v2: -fforce-emit-vtables
    - Sound for C++!

- No link to video yet. link to google doc: https://docs.google.com/document/d/16GVtCpzK8sIHNc2qZz6RN8amICNBtvjWUod2SujZVEo/edit?usp=sharing

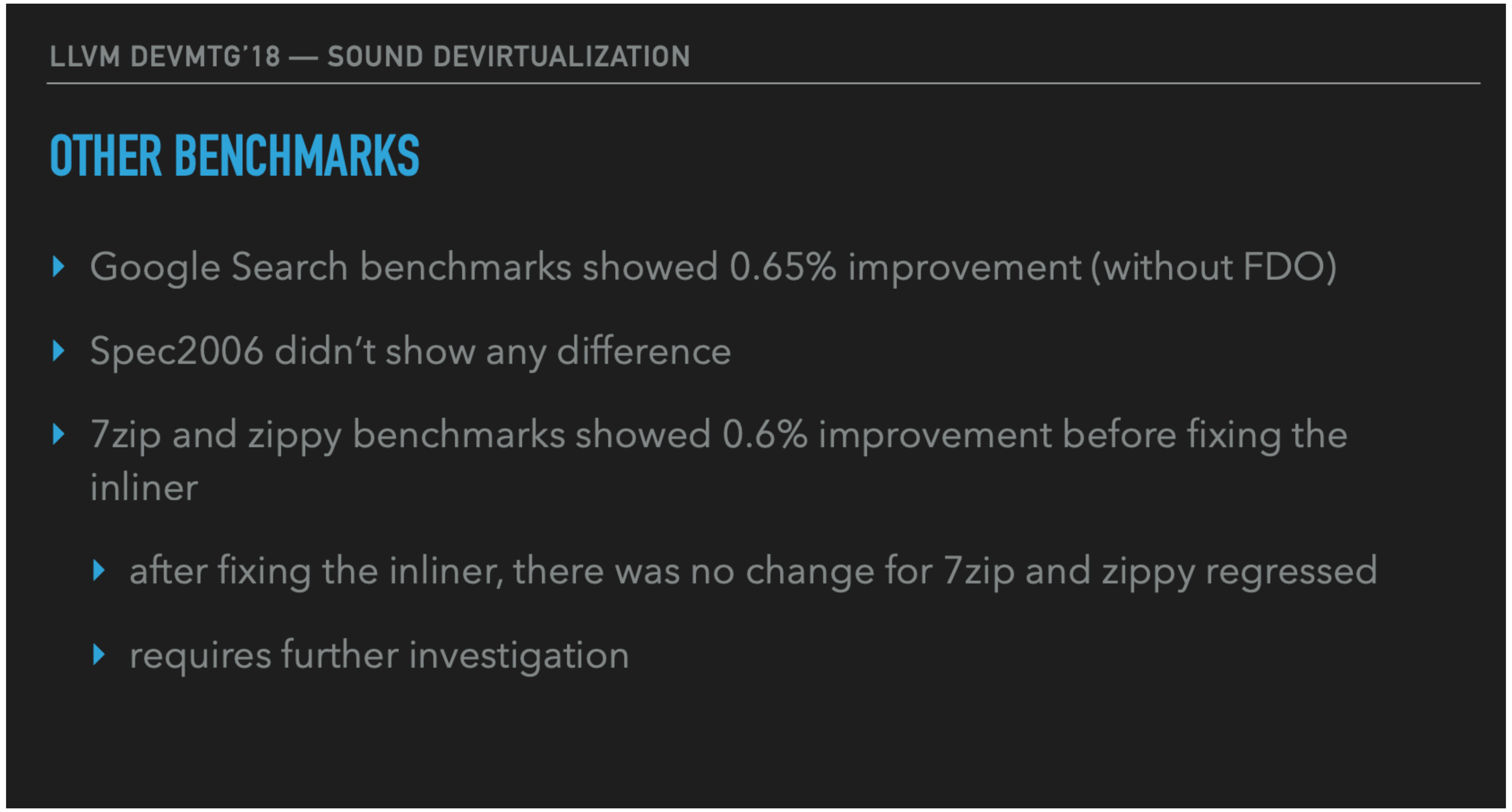
Sound Devirtualization in LLVM
- Piotr Padlewski



LLVM DEVMTG'18 — SOUND DEVIRTUALIZATION

OPTIMIZATIONS STATISTICS (OLD MODEL)

Results for LLVM			
statistic	baseline	devirt	diff
# of vtable loads replaced	1451	14254	882.36%
# of vtable uses devirtualized	982	3269	232.89%
# of vfunction loads replaced	1084	9388	766.05%
# of vfunction devirtualized	954	1861	95.07%
Results for ChakraCore			
statistic	baseline	devirt	diff
# of vtable loads replaced	126	2465	1856.35%
# of vtable uses devirtualized	17	584	3335.29%
# of vfunction loads replaced	45	1082	2304.44%
# of vfunction devirtualized	32	131	309.38%



LLVM DEVMTG'18 — SOUND DEVIRTUALIZATION

OTHER BENCHMARKS

- Google Search benchmarks showed 0.65% improvement (without FDO)
- Spec2006 didn't show any difference
- 7zip and zippy benchmarks showed 0.6% improvement before fixing the inliner
  - after fixing the inliner, there was no change for 7zip and zippy regressed
  - requires further investigation

- Result was impressive
- 0.8% improvement in google infrastructure

Repurposing GCC Regression for LLVM Based Tool Chains - Jeremy Bennett

- GCC test exist in Clang
  - Fork of GCC 4.2 test suite (11 years old)
- Goal: Test a Clang/LLVM tool chain using the latest GCC test suite

- GCC test suite is a mix of C compliance tests, GNU C extension compliance tests, GCC internal tests, C torture tests, as well as regression tests.
- Make GCC test suite more generic and use in Clang

Small topics

- Compiler for machine learning becoming a next trend (Glow: LLVM-based machine learning compiler, Swift for TensorFlow)
- Chandler refusing to upgrade Google infrastructure from libc -> libc++
- LLVM codebase is now C++11, planning to migrate directly to C++ 17! Aiming March 2019
- Nintendo switch compiled with LLVM and LLD!
- C++20 features: Moduels, coroutines..
- LLVM relicensing effort http://llvm.org/foundation/relicensing/

Thank you for your attention!