Using LNT to track performance

ARM

Kristof Beyls

FOSDEM LLVM dev room 2017
5/2/2017

At last year's FOSDEM LLVM dev-room:

Automated Performance-Tracking of LLVM-Generated Code

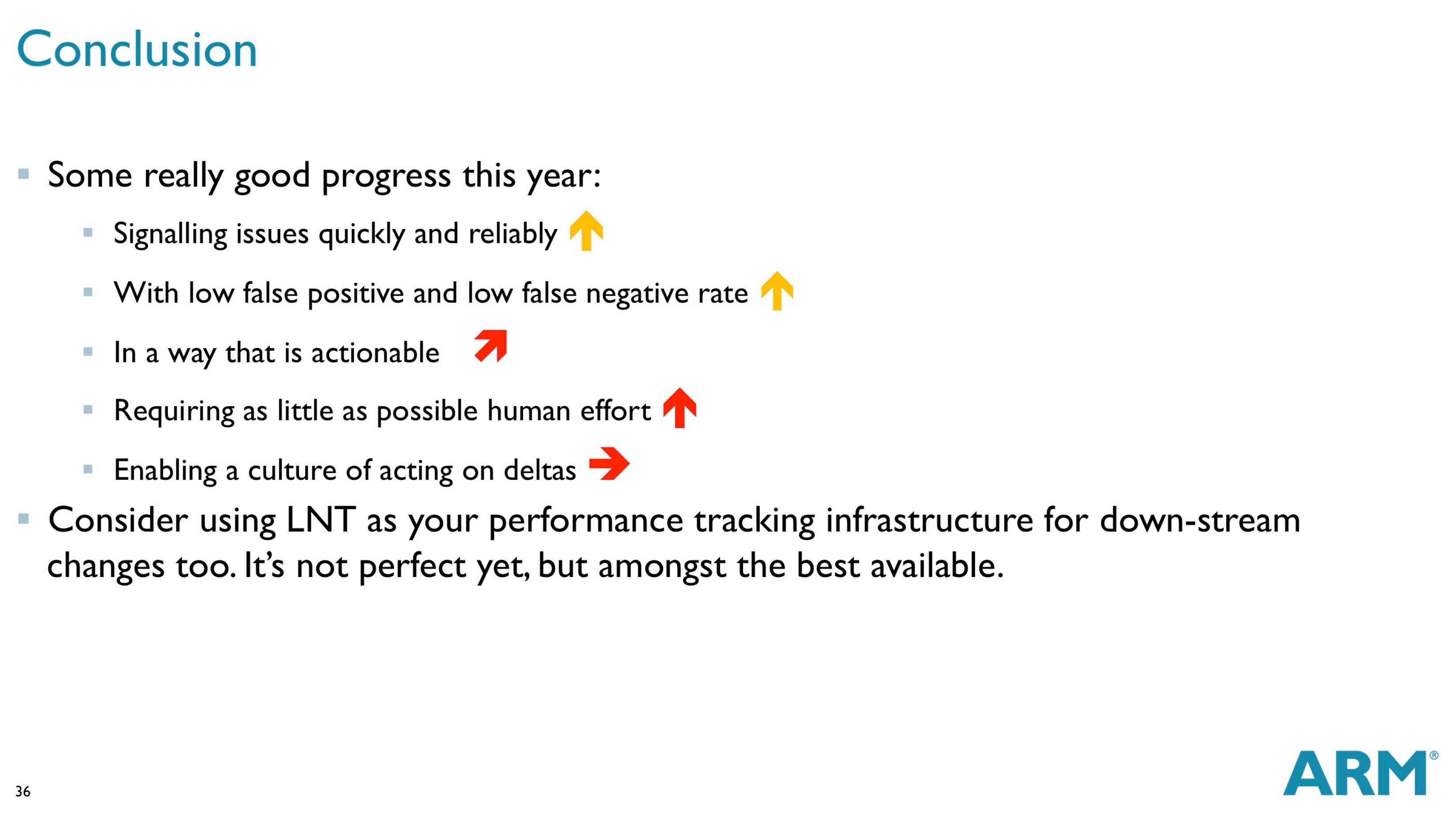
Kristof Beyls
FOSDEM, January 2016

The Architecture for the Digital World®

Last year’s talk mostly

- What was needed in LLVM’s LNT and test-suite subprojects to improve CI performance-tracking of LLVM-generated code.
- A range of ideas for improvements were shown, some implemented, some not.
- We implemented more improvements since, and this presentation aims mainly to demo them.

At last year's FOSDEM LLVM dev-room:

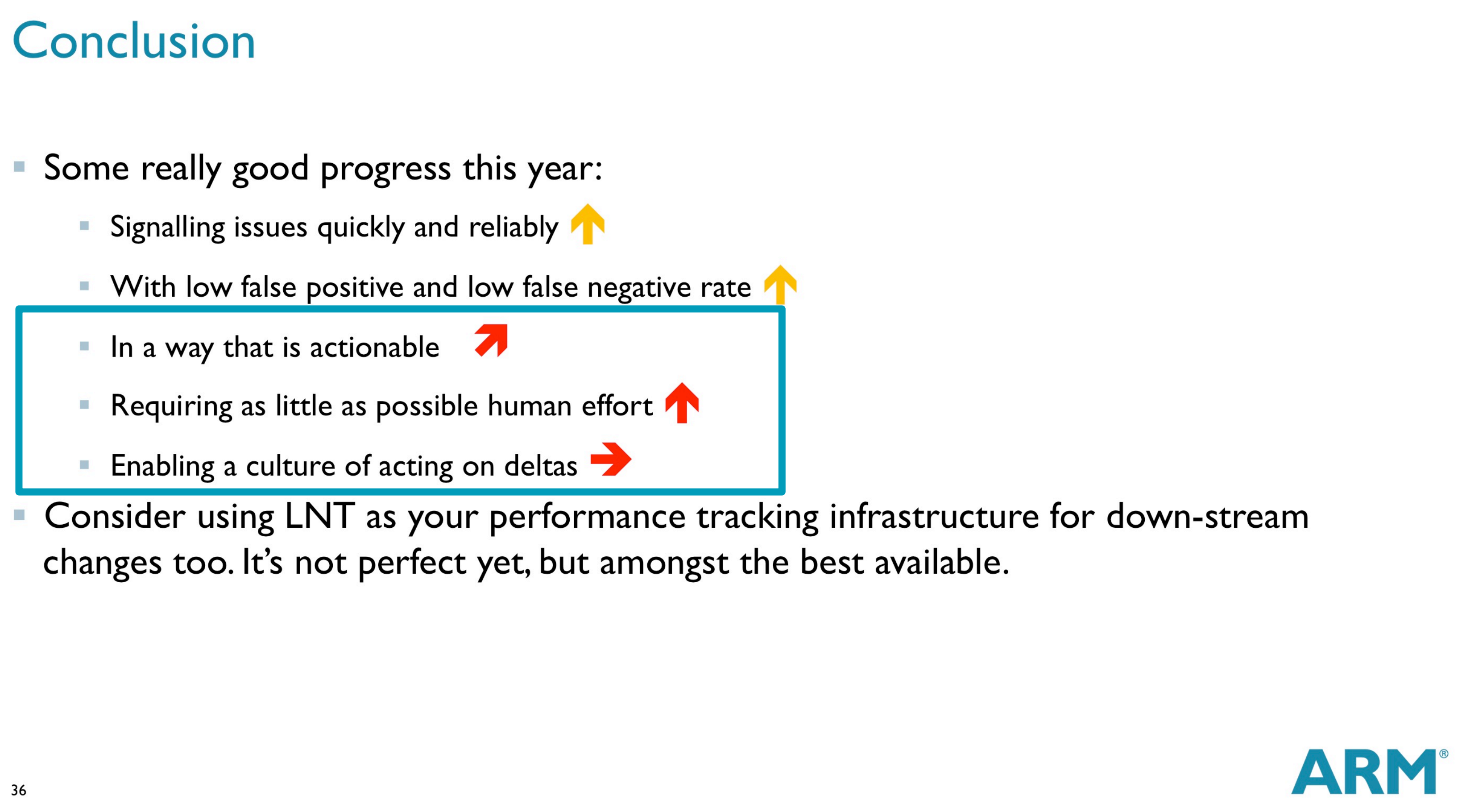


Conclusion

- Some really good progress this year:
  - Signalling issues quickly and reliably ↑
  - With low false positive and low false negative rate ↑
  - In a way that is actionable ↗
  - Requiring as little as possible human effort ↑
  - Enabling a culture of acting on deltas →
- Consider using LNT as your performance tracking infrastructure for down-stream changes too. It's not perfect yet, but amongst the best available.

36

At last year's FOSDEM LLVM dev-room:

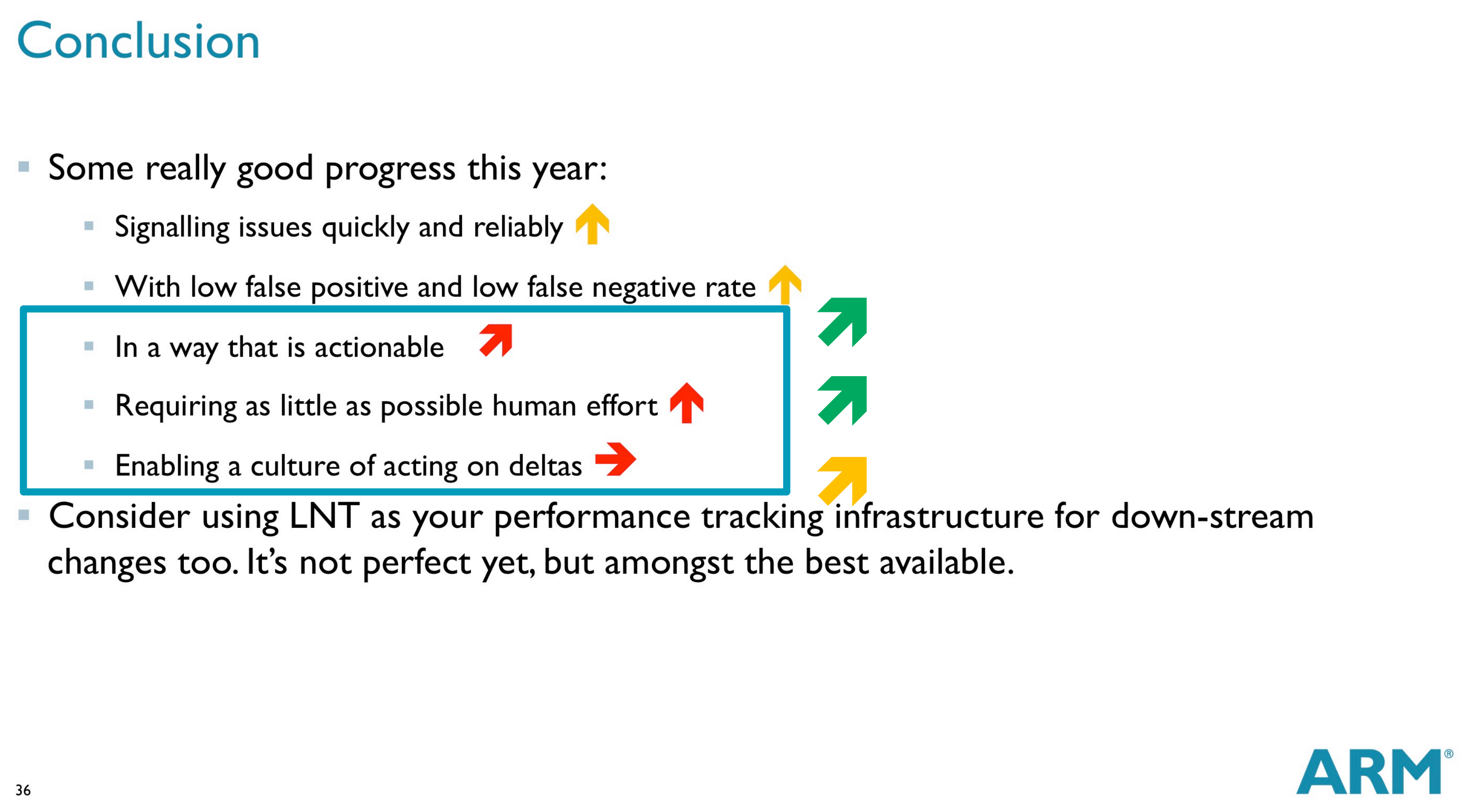


Conclusion

- Some really good progress this year:
  - Signalling issues quickly and reliably ↑
  - With low false positive and low false negative rate ↑
  - In a way that is actionable ↗
  - Requiring as little as possible human effort ↑
  - Enabling a culture of acting on deltas →
- Consider using LNT as your performance tracking infrastructure for down-stream changes too. It's not perfect yet, but amongst the best available.

36

I'll demo a range of improvements



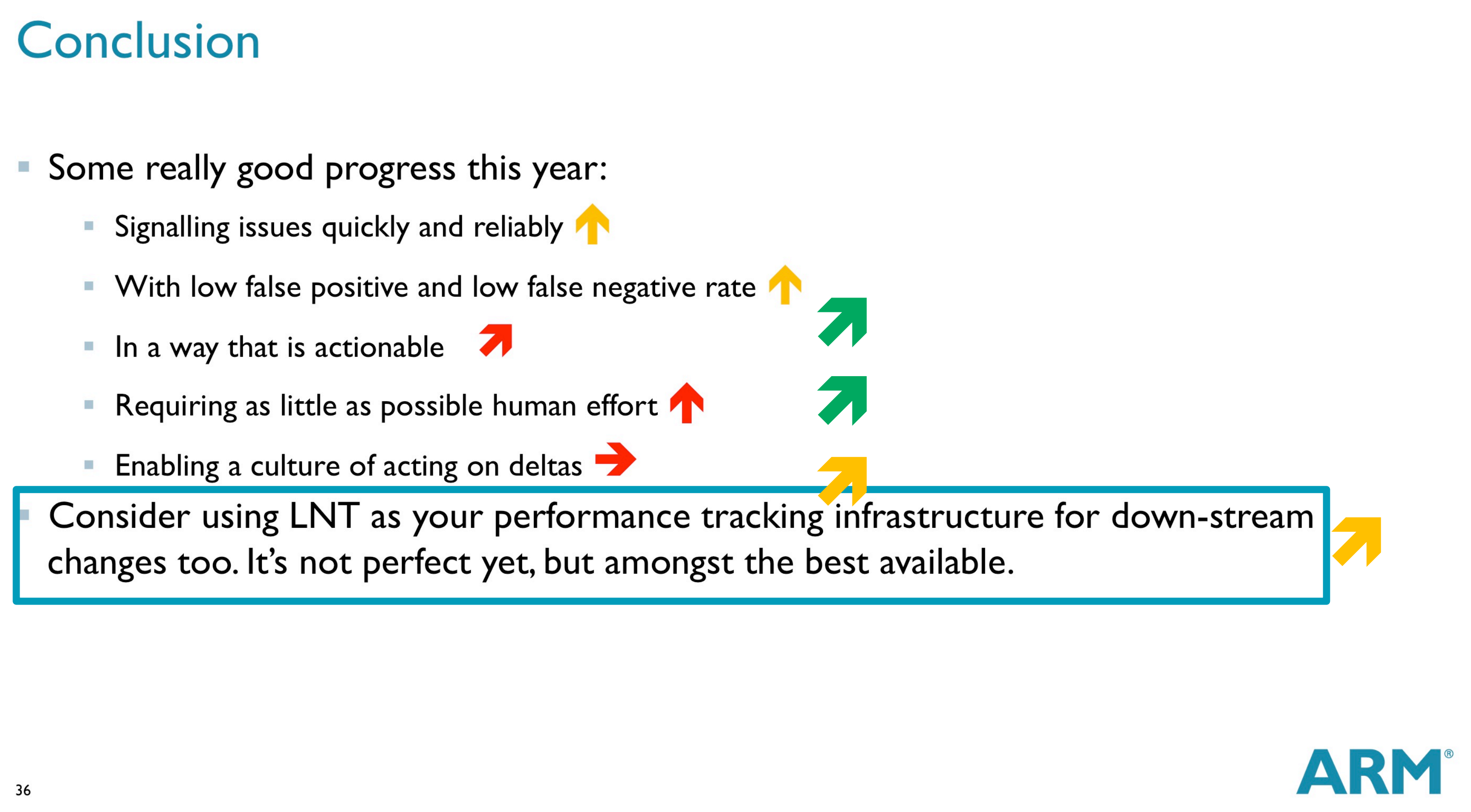
Conclusion

- Some really good progress this year:
  - Signalling issues quickly and reliably
  - With low false positive and low false negative rate
  - In a way that is actionable
  - Requiring as little as possible human effort
  - Enabling a culture of acting on deltas
- Consider using LNT as your performance tracking infrastructure for down-stream changes too. It's not perfect yet, but amongst the best available.

36

ARM®

Also improvements for enabling non-LLVM projects



Conclusion

- Some really good progress this year:
  - Signalling issues quickly and reliably
  - With low false positive and low false negative rate
  - In a way that is actionable
  - Requiring as little as possible human effort
  - Enabling a culture of acting on deltas
- Consider using LNT as your performance tracking infrastructure for down-stream changes too. It's not perfect yet, but amongst the best available.

36

In a way that is actionable ↗

Requiring as little as possible human effort ↗

ARM

DEMO

Next, use llvmlab bisect

- All clang binaries built by a fast builder stored in a "build cache".
- The llvmlab bisect utility can bisect issues, using these cached binaries https://github.com/llvm-mirror/zorg/tree/master/llvmbisect/llvmlab
- Enables finding commit that caused a regression quickly.
- Documentation: https://github.com/llvm-mirror/zorg/blob/master/llvmbisect/docs/llvmlab_bisect.rst

```
bash-3.2# ~admin/zorg/utils/llvmlab bisect --max-rev
131837 ./test.sh
no
FAIL: clang-r131837-b8165
no
FAIL: clang-r131835-b8164
no
FAIL: clang-r131832-b8162
no
FAIL: clang-r131828-b8158
yes
PASS: clang-r131795-b8146
no
FAIL: clang-r131809-b8151
no
FAIL: clang-r131806-b8149
no
FAIL: clang-r131801-b8147
clang-r131795-b8146: first working build
clang-r131801-b8147: next failing build
```

Bisect, points to specific commit

- Sharing this info with original author by replying to the commit email on llvm-commits:
  I have been bisecting some big performance regressions on SingleSource/Benchmarks/BenchmarkGame/Large/fasta and SingleSource/Benchmarks/Misc-C++/bigfib (see http://llvm.org/perf/db_default/v4/nts/daily_report/2016/4/14?filter-machine-regex=aarch64|arm|green&num_days=7), and it points to this commit.
  Would you mind having a look at those ?
- Regression got fixed within 48 hours
- Regressions are cheap to fix if detected quickly.

ARM

Enabling a culture of acting on deltas

Further enabling culture of acting on deltas

- Have more public performance-reporting bots
- Upgrade the public LNT server to the latest version of LNT
- Improve coverage for more use cases in test-suite. E.g.
  - in the past year, bitcode files produced by Halide were added
  - Benchmarks representing HPC are starting to be added
  - But needs to be done in a way such that test-suite doesn't take longer to run, otherwise we loose revision resolution.
- Automated emails on performance regressions?
  - With automated bisect to a single commit?
  - With automated highlighting of hot basic block code changes?

ARM

Use in non-LLVM projects

ARM

Use in non-LLVM projects

- The interface (a JSON file) to use LNT on non-LLVM-test-suite benchmarks has been documented for this, see http://lnt.llvm.org/importing_data.html.

- At ARM:
  - LLVM team uses LNT.
  - GCC team also uses LNT.
  - Cycle Models team, developing a Verilog-to-C++ compiler also uses LNT.

- For LLVM test-suite benchmarks: the cmake/lit-ification made it easy to plug in extra benchmarks, and e.g. perf profiling working out of the box without extra work.

Summary

- Since last year, major improvements in LNT to help with a CI setup to monitor the speed of generated code:
  - In a way that is actionable↗
  - Requiring as little as possible human effort↗
  - Enabling a culture of acting on deltas↗
- Documentation has improved to make LNT easier to integrate with non-LLVM projects.
- Please give feedback when using LNT, we'd like to learn from your experience, either via email or via https://llvm.org/bugs/, component LNT.

ARM