EuroPython 2023

Diego Russo

20th of July, 2023

Agenda

- Introduction
- Python on AArch64
- Python on Windows on Arm
- Python and Machine Learning
- What’s next?

Who Am I?

- Diego Russo
- I’ve been using Python since 2006
- Other expertise
  - ML, Shell, Linux, Automation, other programming languages, etc..
- 12 years at Arm Ltd
  - Principal Software Engineer
  - ML group since 2020
  - Since April 2023, secondment in the runtime team in the OSS group
  - I’ve been running a Python Guild with more than 920 members for 3 years
- Secondment objective: **Investigate CPython on Arm platforms**

What does Arm do?

- Arm **designs the technology building blocks** that traditional semiconductor companies use to create silicon chips for a wide range of devices, applications and services.
- Business model based on **license** and **royalties**
- Trusted technology for a **vast ecosystem**

1000+ Ecosystem Partners

Arm Open-Source Software Group

Supporting your needs around firmware, operating systems, runtimes, compilers, middleware, workloads and libraries.

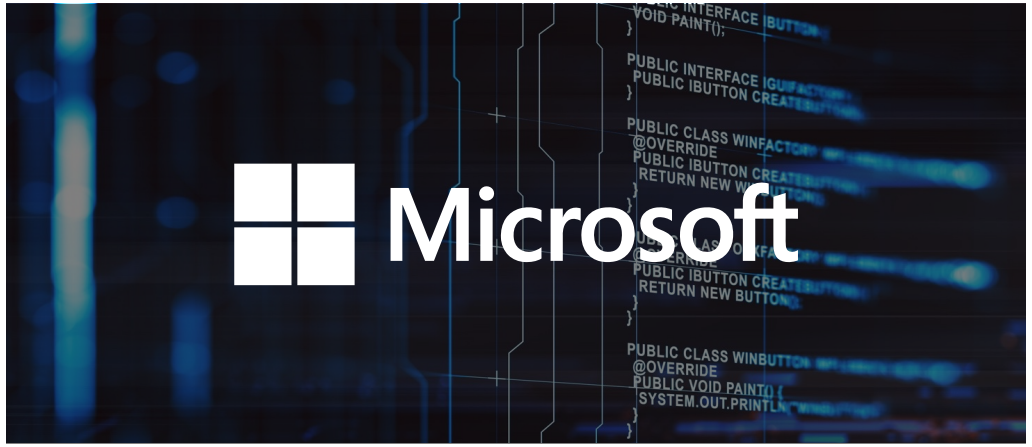
- **1 in 3** Arm engineers dedicated to software
- Over **130,000** people-days per year devoted to over **1000** open-source projects
- Over **100m** active users of open-source libraries such as Arm NN for machine learning.

Arm architecture is all around us

From chip to cloud

Clouds everywhere are deploying Arm-based servers

Next Generation Native Experiences for Windows on Arm



Python on AArch64

Python on AArch64

Enablement effort from upstream community

Arm related commits since 2001

First AArch64 commit in August 2014

CPython is written in “portable” C

Changes are more related to tooling (build, installation, etc…)

PEP599 - The manylinux2014 Platform Tag

Created in April 2019 and accepted in July 2019

Extends manylinux2010 platform tag

It officially introduces the support for Arm platforms

- Armv7 (Armv7l)
- Armv8 (AArch64)

This platform tag has been superseded by PEP600

E.g: SQLAlchemy-2.0.16-cp311-cp311-manylinux_2_17_aarch64.**manylinux2014_aarch64**.whl

Python on AArch64

Enablement effort from Arm and partners

Enable AArch64 wider ecosystem(s) for the first public clouds that had Arm instances

- Not just Python ecosystem

Have the same smoother experience of other main platforms

- If pip doesn't find the built distribution for the running platform, it reverts to the **source distribution** and try to compile the package

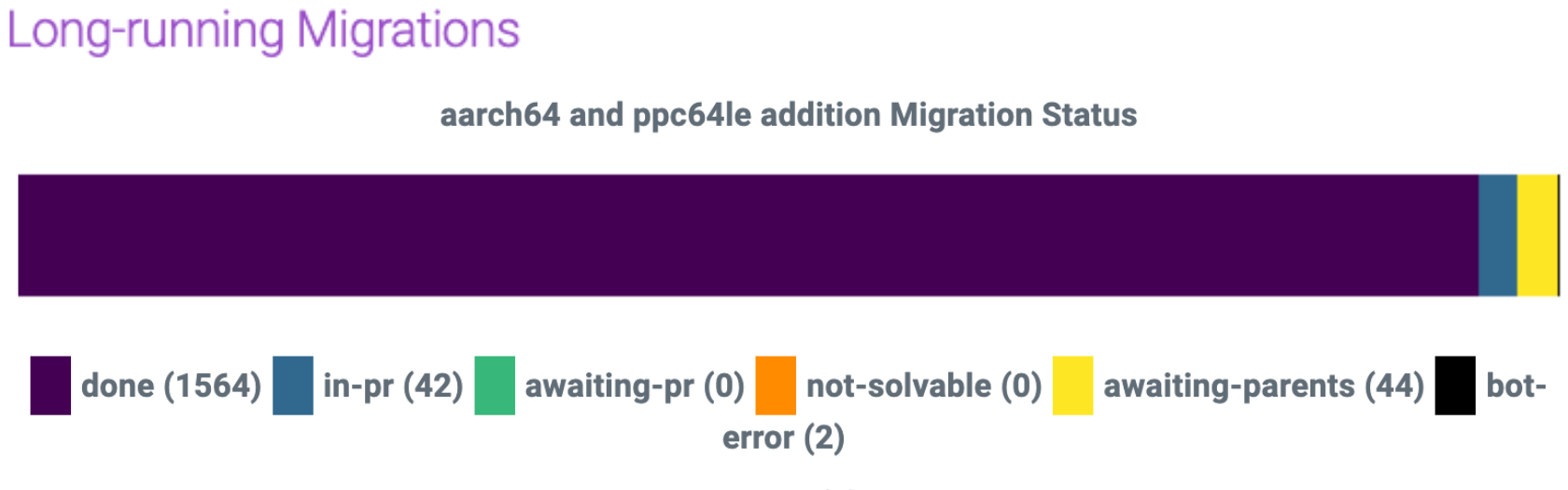
Analyzed 2920 packages present in PyPi

- 2500 Python packages and 420 wheels
- Install and test them on x86 and AArch64
- Sorted in a priority list, the ones failing on AArch64 have been tackled
  - Enable the package to be built on AArch64, verify it, get accepted by the maintainer and move to the next one
  - Fixes related to both applications and/or build system in order to produce a built distribution for AArch64.
  - Avoid packages where Arm has already a long-term engagement (e.g., TensorFlow)
- Started in 2020, after 2 years more than 200 Python project have been enabled by generating aarch64 built distributions

Python on AArch64

Conda Forge

- Acted only on a dozen packages
- Historical data:
  - 2020-05: **531** Done
  - 2021-05: **971** Done
  - 2022-09: **1311** Done
  - 2022-07: **1564** Done



Source: https://conda-forge.org/status/#aarch64andppc64leaddition

Python on AArch64

What about performance?

https://speed.python.org/ doesn't have AArch64 benchmarks

Software stack

- Pyperformance: https://github.com/python/pyperformance
- Codespeed: https://github.com/python/codespeed/

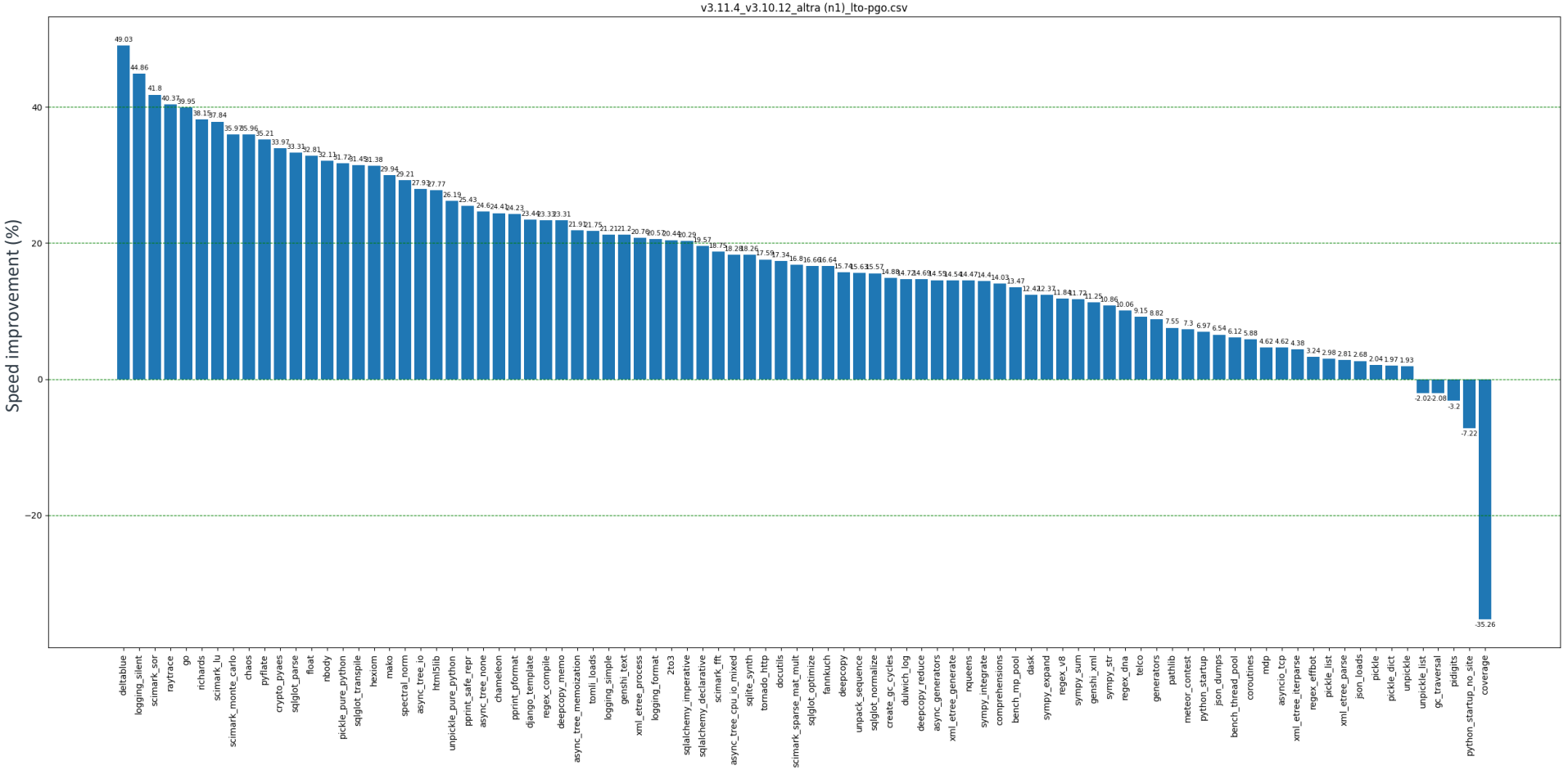
Replicated internally

- Branches 3.10, 3.11 and main
- Different sources (Ubuntu, Deadsnakes, compiled)
- GCC –mcpu=native optimization
- Neoverse-N1/N2/V1, Cortex-A57/72, x86, AWS Graviton 2, Graviton 3, etc..
- Multi-core vs single-core

There is an ongoing upstream engagement to make AArch64 performance metric public… stay tuned.

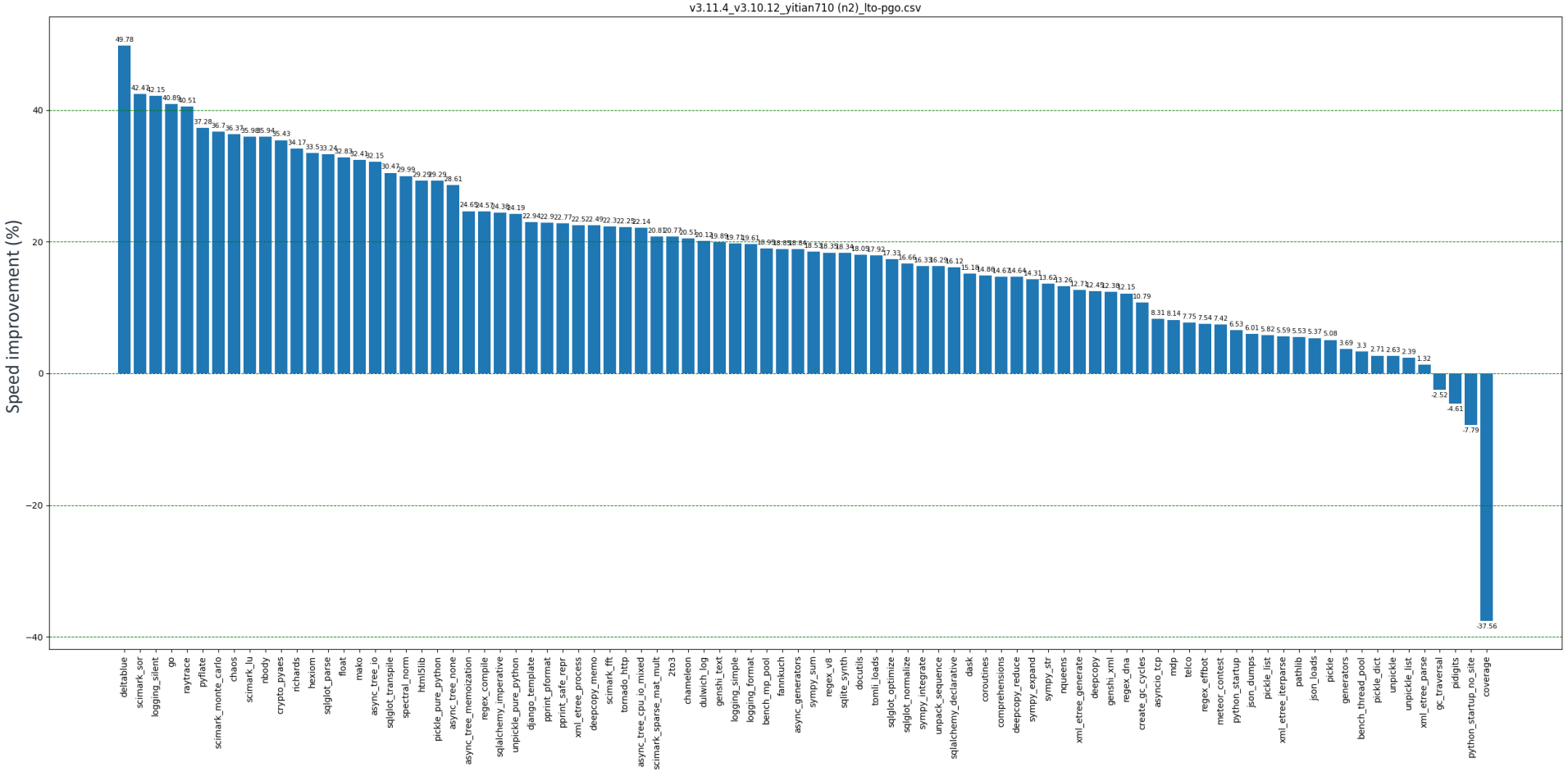
Python on AArch64

3.11 vs 3.10 on Ampere Altra (Neoverse-N1)



Python on AArch64

3.11 vs 3.10 on Alibaba Yitian710 (Neoverse-N2)



Python on AArch64

Distribution channels

Python can be distributed via multiple channels

- Distro packages (E.g.: Ubuntu), Deadsnakes, compiled from sources

Noticed differences between binaries

- PIE (Position-Independent Executable): depend on the distribution
- Different effects depending on the platform (aarch64, x86)

Deadsnakes

- Overall deadsnakes binaries are faster (1% - 15%) compared to vanilla compiled source code
- bench_mp_pool though is 75% slower than the vanilla compiled version
- Deadsnake compiled is slightly faster than the ppa binaries

Conclusion

- Don’t assume all binaries are the same
- Channels use different flags (-flto), environments and patches

Python on AArch64

GCC –mcpu=native optimization

All packages compiled with GCC are compiled against Armv8.0-A (aarch64)

Armv8.1-A to Armv8.9-A and Armv9.0-A to Armv9.4-A

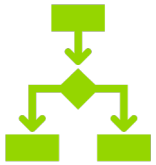
The main reason is backward compatibility and portability across platforms

What if we recompile CPython against more recent CPUs?

Neoverse-N1: Armv8.2-A

Neoverse-V1: Armv8.4-A

Neoverse-N2: Armv9.0-A



What version of GCC?

GCC11, GCC12, GCC13

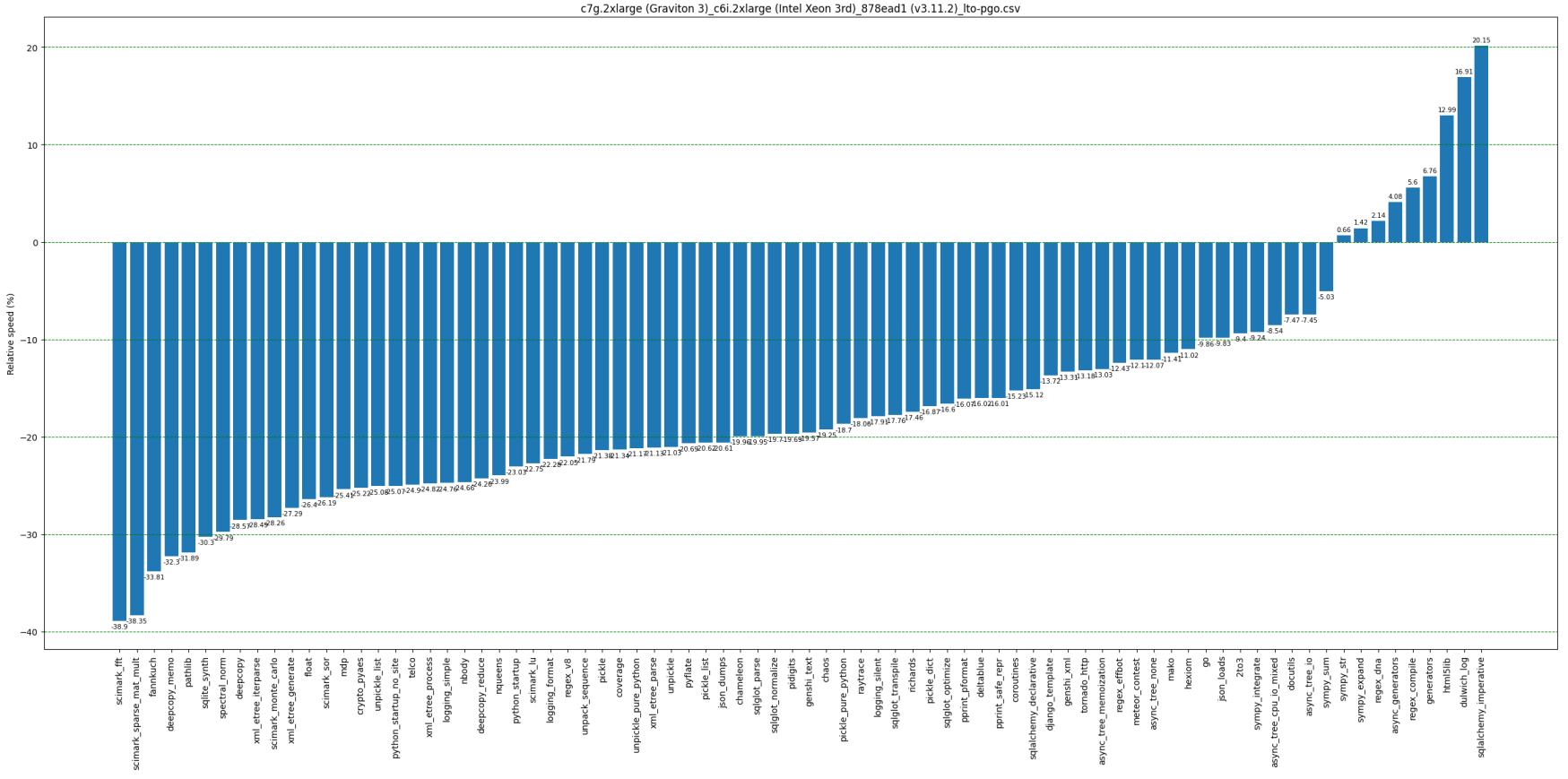
Using -mcpu=native when host compiling CPython doesn't bring any real benefits

The only performance improvement noticed is between gcc11 -mcpu and gcc12 -mcpu on V1 (up to 10% faster)

Python on AArch64

AWS benchmark results for single process

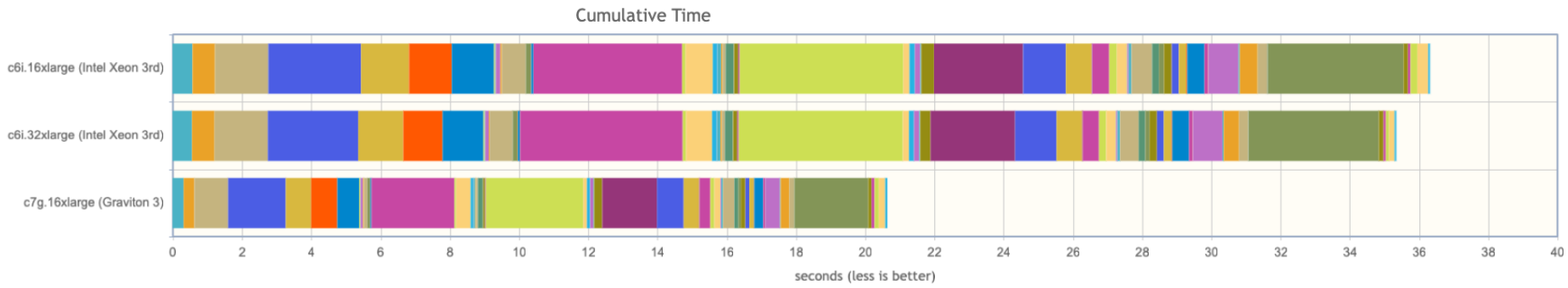
- Run on a variety of flavors
  - C7g: Graviton 3
  - C6i: Intel Xeon 3rd gen
- Graviton 3 is more cost effective for the whole benchmark workload compared to the x86 equivalent
- Similar behaviors across CPython versions



Python on aarch64

AWS benchmark results for multi processes

- Run an instance of `pyperformance` for every CPU
- Run it for longer
- Used the biggest flavors available
- Graviton 3 took almost half of the time



Python on Windows on Arm (WoA)

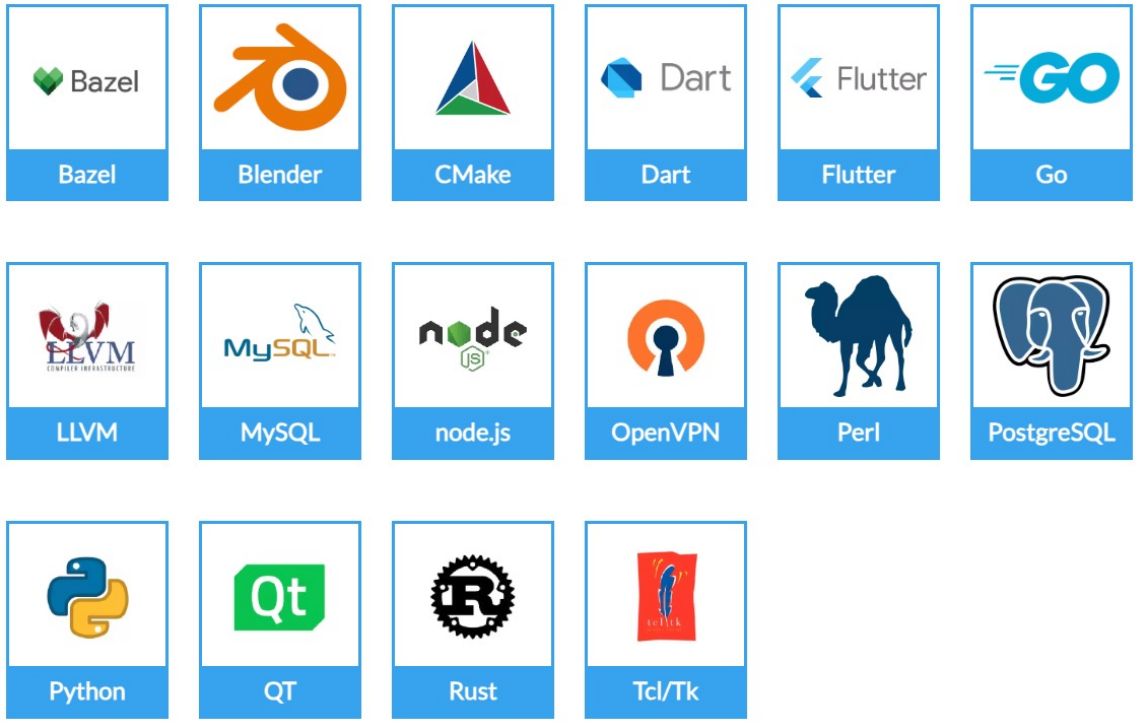
Python on Windows on Arm (WoA)

Enablement

- Python 3.8 added experimental support for WoA
  - Download it via NuGet or build it from source
- From Python 3.11 WoA is **officially** supported
- Joint effort to build a WoA ecosystem which supports native development

Linaro

arm Qualcomm Innovation Center Microsoft CIX



arm

Python on Windows on Arm (WoA)

Enablement

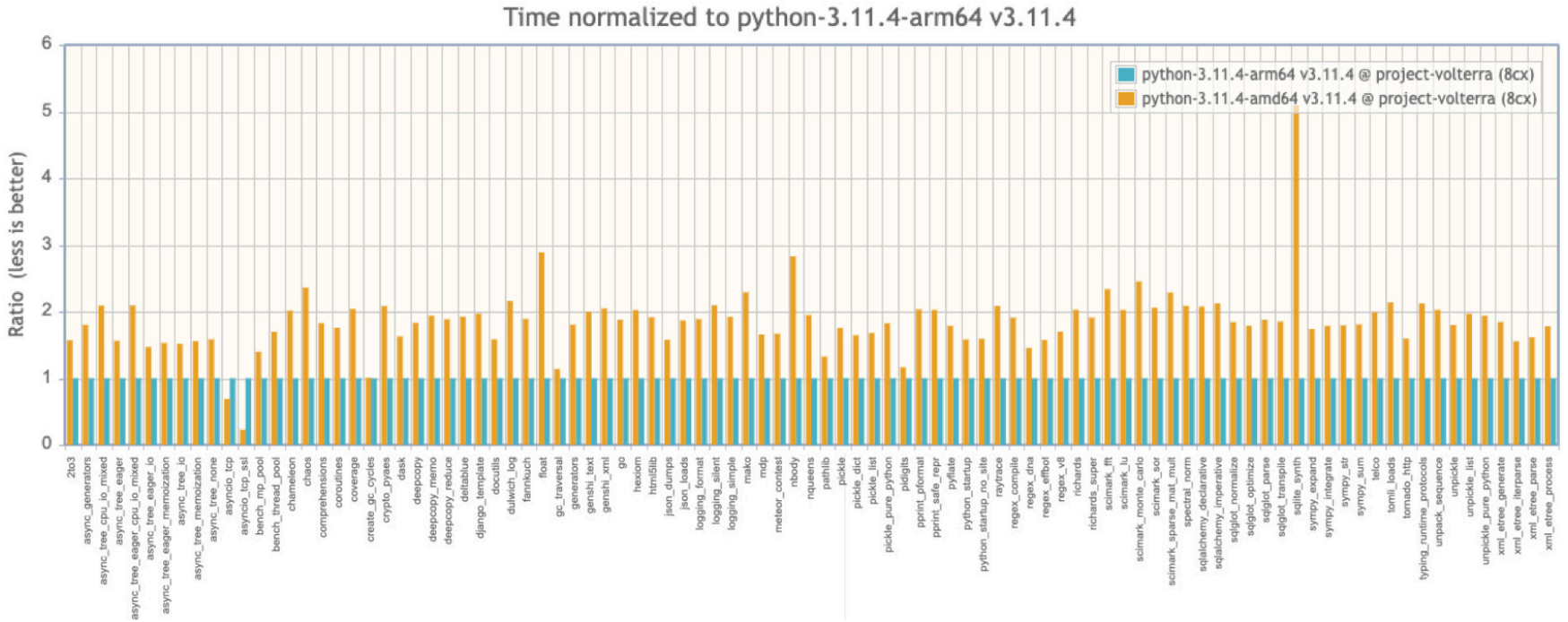
Tested the top 520 packages: over 70% succeeded	• Acted on a subset: setuptools, pip, distlib, numpy, cffi, lxml, greenlet, pytorch, etc… • But also on third party libraries: OpenBLAS, FreeType, libpng, zlib, ZeroMQ, etc… • And Toolchains: Ninja, CMake, Meson, Bazel, etc…
Linaro hosts a buildbot worker for building Python on WoA	• It's a Surface Pro X
Collaborative space for Python enablement on WoA	• All feedback, contribution, and findings are welcome!

Do you want to build and test your package for WoA?	• If you have an WoA supported device: Building Native Windows on Arm Apps with Python
	• If you don't have an WoA supported device
	  • Build: cross compilation is still the best approach
	  • Test: QEMU + Wine + Docker to the rescue

Python on Windows on Arm

Performance

- Windows 11 on Arm can run x64 binaries via emulation
- For pyperformance, Python Arm64 is almost twice as fast (on average) than amd64 (emulated) version
- **Recommendation**: start building native packages for this platform



Python and Machine Learning

Python and Machine learning

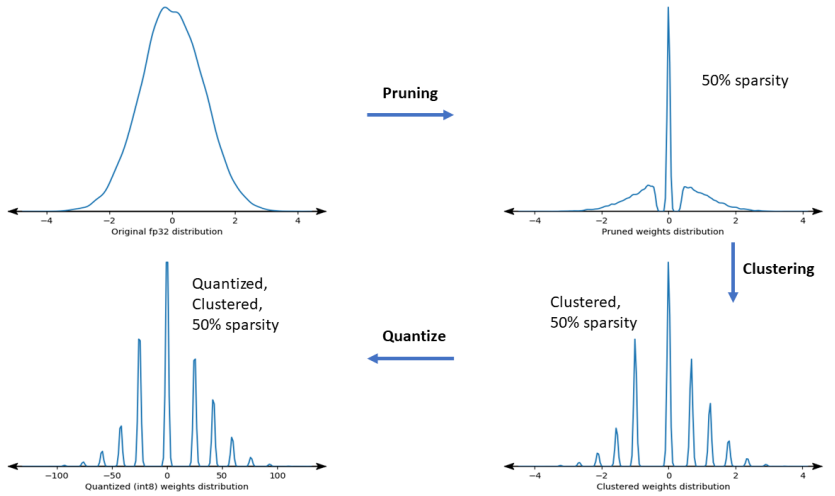
TensorFlow Lite and Model Optimization

TensorFlow Lite

- Introduction of int16 post-training quantization mode for activations and corresponding reference kernels
- Continuous improvements in quantization support
- Maturing TFLite-to-TOSA legalization code
- Engagement around 4-bit support in TFLite

Model Optimization

- Contributing weight clustering optimization technique
- Contributing support for applying optimizations collaboratively
- Several blogposts on model optimizations



Python and Machine learning

TensorFlow

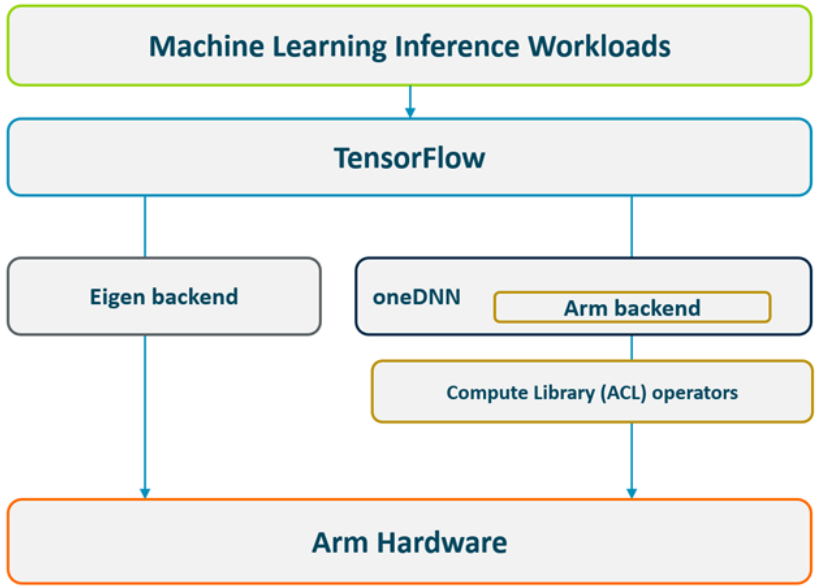
AArch64 packages are available from TensorFlow 2.10

Collaboration between AWS, Arm, Google and Linaro

For bleeding edge builds, Arm provides Docker images

Both Eigen and oneDNN+ACL backends

Guide available for building TensorFlow for WoA



Python and Machine learning

PyTorch

AArch64 packages are available from PyTorch 1.8.0

For bleeding edge builds, Arm provides Docker images

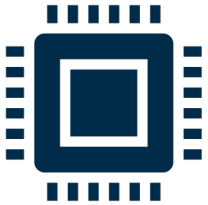
- Both OpenBLAS and oneDNN+ACL backends

Unfortunately, there is no PyTorch package for WoA

- Linaro is tracking its status

Python and Machine learning

Other contributions



Apache TVM

Support to Arm Compute Library

tvmc command line and Python API

Packaging for various Python versions and platforms

Operator support for Tensorflow and TFLite

Support for many microcontroller platforms

OpenBLAS

Tweaks for Neoverse cores

Enabled SVE for Neoverse-V1

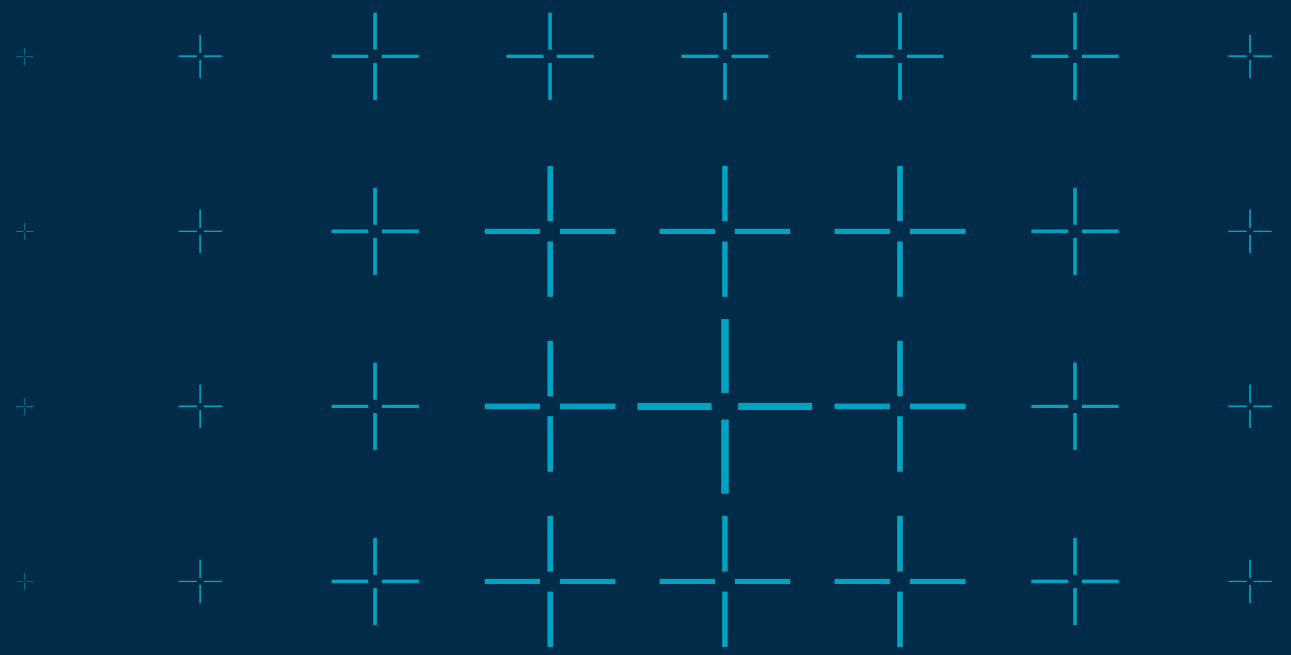
Implemented dot product with SVE

NumPy

Implemented some faster math routines

arm

What’s next?

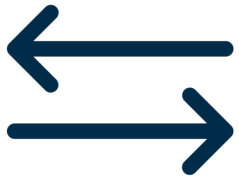


What gaps do we have today?

AArch64 should be moved from Tier2 to Tier1 of the platform support	• The ecosystem is healthier and almost on par with x86
aarch64/arm64 not present on the Speed Centre	• I'm personally working on this
Windows on Arm ecosystem work is just getting started	• No TensorFlow and Pytorch packages for WoA • Install Python on WoA
Performance on AArch64 are great but there is always room for improvement	• Improve single process performance

What’s next?

How you can help to get Python and its ecosystem even better on Arm devices



Start migrate your workloads to Arm.

If you encounter any issue, engage with upstream communities

If you provide a native package, start building and testing it on Arm platforms

Don’t rely on emulation

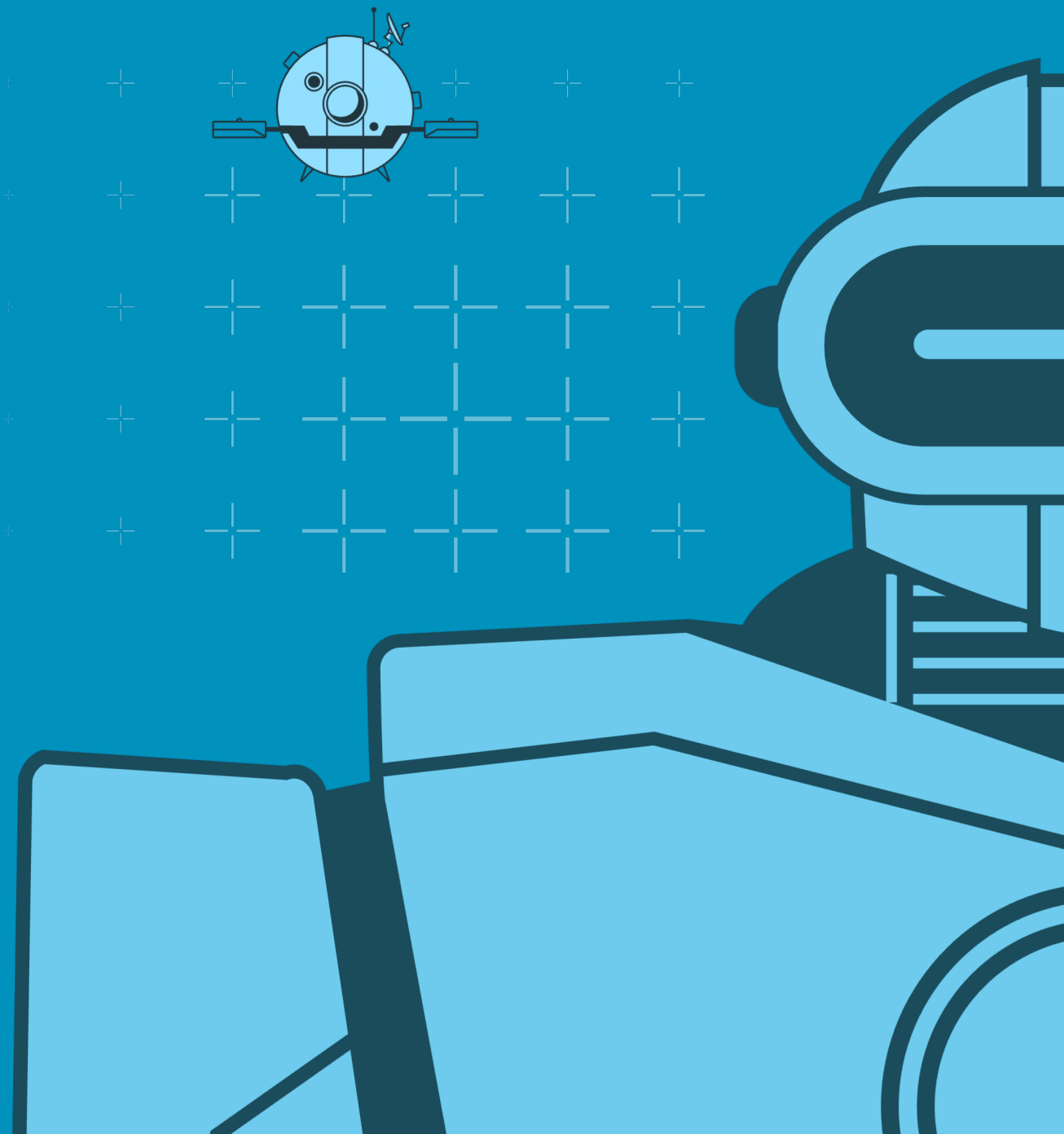
Engage directly with us…

arm Developer Program

The community to build your future on Arm

- Get fresh insights directly from Arm experts
- Connect with like-minded developers
- Build on your expertise and become an Arm Ambassador

https://www.arm.com/developerprogram



arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה