Challenge Accepted:

Python Packaging Infrastructure for the RISCV64 Ecosystem

RISC-V® EU Summit 2025, Paris

RISE
RISC-V Software Ecosystem

Agenda

Intro

Using Python on RISCV64

The **wheel_builder** Project

Challenges: Our Experience (So Far)

Conclusion

Intro

- Collaborative project for accelerating open-source development on RISC-V, improving software quality
- Many specialized working groups
- More info: https://riseproject.dev/

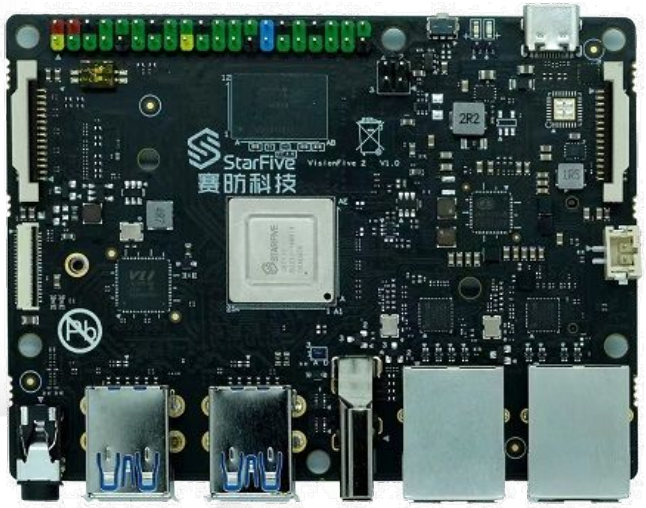
- Services company specializing in all things embedded - e.g. the Linux Kernel, Yocto, U-Boot, Zephyr, hardware design, compilers, toolchains
- Based in Nice, France, with a global presence
- More info: https://baylibre.com/

- Bringing scalable, high-performance AI solutions to the Data Center
- Active member of the RISC-V community, involved in both hardware and software ecosystems
- More info: https://www.rivosinc.com/

Using Python on RISCV64

Device Setup

- **VisionFive 2** (https://www.starfivetech.com/en/site/boards)
- **Ubuntu 24.04.2 LTS** (https://ubuntu.com/download/risc-v)
  - apt update && apt -y upgrade already done
  - Also installed cmake, ninja-build, autoconf, python3-pip
- **System Python:** 3.12.3
- Connected via SSH or serial port
- Python virtual environment prepared using **python3 -m venv venv && source venv/bin/activate**



Let's Install **NumPy** (and time it)

```
(venv) ubuntu@ubuntu:~/sandbox/python$ time pip install numpy
Collecting numpy
  Using cached numpy-2.2.4.tar.gz (20.3 MB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: numpy
  Building wheel for numpy (pyproject.toml) ... done
  Created wheel for numpy: filename=numpy-2.2.4-cp312-cp312-linux_riscv64.whl
  Stored in directory: /home/ubuntu/.cache/pip/wheels/02/87/cb/0e40c75232acfb
Successfully built numpy
Installing collected packages: numpy
Successfully installed numpy-2.2.4

real    17m29.615s
user    49m19.049s
sys     1m46.333s
```

For comparison, the same build on an emulated system with Ryzen 9 7900 + 64GB of RAM took **~8 minutes**

Let's Install **matplotlib** (and time it)

```
(venv) ubuntu@ubuntu:~/sandbox/python$ time pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.10.1.tar.gz (36.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 36.7/36.7 MB 2.1 MB/s eta 0:00:00

  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.1.tar.gz (13.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.5/13.5 MB 3.2 MB/s eta 0:00:00

  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib)
  Downloading fonttools-4.57.0-py3-none-any.whl.metadata (102 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.5/102.5 kB 399.6 kB/s eta 0:00:00
```

...

```
      File "/tmp/pip-build-env-o3cm09gk/overlay/lib/python3.12/site-packages/setuptc
        exec(code, locals())
      File "<string>", line 1048, in <module>
    RequiredDependencyException:

    The headers or library files could not be found for jpeg,
    a required dependency when compiling Pillow from source.

    Please see the install instructions at:
       https://pillow.readthedocs.io/en/latest/installation/basic-installation.html


    [end of output]

  note: This error originates from a subprocess, and is likely not a problem with pip.
  ERROR: Failed building wheel for pillow
Successfully built matplotlib contourpy kiwisolver
Failed to build pillow
ERROR: Could not build wheels for pillow, which is required to install pyproject.toml-

real    11m37.607s
user    29m47.046s
sys     1m16.197s
```

Many dependencies required… but Pillow fails to build. We could work around this - but should we? Poor user experience vs other architectures, e.g. x86_64

Why So Long? How **pip** Works

- Python package installer, e.g. **pip install numpy**
- Searches index for package (by default, https://pypi.org/) **matching your host architecture**, or a pure Python wheel (if available)
- If it can't find a match, it downloads a source distribution, e.g. **numpy-2.2.4.tar.gz**, and then **tries to compile it**
  - **No riscv64 binary wheels on PyPI**, so pip will do this for every package without a pure Python wheel (**see:** https://discuss.python.org/t/packaging-support-for-riscv64/58475)
  - **How do we know that this built version is optimized the same way?**
  - Packages like NumPy have dependencies, e.g. OpenBLAS, **and use specific versions to build specific releases** - ours might not even include it (which version did you install?) - **and therefore could have different behavior!**

The Manual Build Dilemma

- NumPy installed “quickly” - **but it’s bottom-of-stack**
  - matplotlib needs extra effort
- **Alternatives:**
  - Stick to Python stdlib + pure Python wheels - limited workloads
  - Try to build packages manually - lots of extra setup/debug
- Either way, **major deviation from standard Python experience**
- **See also:** https://fosdem.org/2025/schedule/event/fosdem-2025-5659-towards-seamless-python-package-installation-on-riscv64/
- **What can we do?**

The **wheel_builder** Project

What is **wheel_builder**?

- **https://gitlab.com/riseproject/python/wheel_builder**
- **Goals:**
  - **Lay groundwork for default riscv64 support**
  - **Enable LLM/deep learning applications**
- **Package registry** offering **tested** wheels for **Python 3.10 to 3.13**
  - **Also includes documentation** (https://riseproject.gitlab.io/python/wheel_builder/)
- Under the hood:
  - GitLab CI pipelines
  - Architecture- and automation-specific patches
  - Emulated (**qemu-user-static**), containerized RISCV64 builds (customized **manylinux)**
  - **cibuildwheel** (with some tweaks), **auditwheel**
  - Paired with mirrors of upstream repos (also maintained by RISE)

Example: vLLM: A Look at Dependencies

```
vllm
├── aiohttp [required: Any, installed: 3.11.9]
│   │   └── frozenlist [required: >=1.1.0, installed: 1.5.0]
│   ├── frozenlist [required: >=1.1.1, installed: 1.5.0]
│   ├── multidict [required: >=4.5,<7.0, installed: 6.1.0]
│   ├── propcache [required: >=0.2.0, installed: 0.2.1]
│   └── yarl [required: >=1.17.0,<2.0, installed: 1.18.3]
│       ├── multidict [required: >=4.0, installed: 6.1.0]
│       └── propcache [required: >=0.2.0, installed: 0.2.1]
├── fastapi [required: >=0.107.0,!=0.114.0,!=0.113.*, installed: 0.115.5]
│   ├── pydantic [required: >=1.7.4,<3.0.0,!=2.1.0,!=2.0.1,!=2.0.0,!=1.8.1,!=1.8, installed: 2.10.2]
│   │   ├── pydantic_core [required: ==2.27.1, installed: 2.27.1]
├── lm-format-enforcer [required: >=0.10.9,<0.11, installed: 0.10.9]
│   ├── pydantic [required: >=1.10.8, installed: 2.10.2]
│   │   ├── pydantic_core [required: ==2.27.1, installed: 2.27.1]
├── mistral_common [required: >=1.5.0, installed: 1.5.1]
│   ├── pydantic [required: >=2.6.1,<3.0.0, installed: 2.10.2]
│   │   ├── pydantic_core [required: ==2.27.1, installed: 2.27.1]
│   ├── requests [required: >=2.0.0,<3.0.0, installed: 2.32.3]
│   │   ├── charset-normalizer [required: >=2,<4, installed: 3.4.0]
```

Need to support all of this (and more!)

wheel_builder Binary Package List - March, 2025

Package	Rank	Package	Rank
charset-normalizer	1	sqlalchemy	16
numpy	2	**frozenlist**	17
pyyaml	3	grpcio	18
cryptography	4	greenlet	19
cffi	5	**pillow**	20
protobuf*	6	**propcache**	21
pandas	7	**scipy**	22
markupsafe	8	**rpds-py**	23
aiohttp*	9	pynacl	24
pydantic-core	10	**lxml**	25
wrapt	11	**msgpack**	26
pyarrow*	12	coverage	27
yarl	13	psycopg2-binary	28
psutil	14	**regex**	29
multidict	15	bcrypt	30

Table 1: Top 30 PyPI Packages for Python 3.10, March 2025

- Packages in **bold** are currently supported, while asterisks (*) are planned
- Built using GitLab CI pipelines that closely mirror upstream repos, e.g. any local changes submitted there first (if possible)

How does this improve the user experience?

Let's Install NumPy - Now with **wheel_builder**

- Just set **PIP_INDEX_URL** and run **pip (24.1 or newer)**
  - (PIP_INDEX_URL=https://gitlab.com/api/v4/projects/riseproject%2Fpython%2Fwheel_builder/packages/pypi/simple):

```
(venv) ubuntu@ubuntu:~/sandbox/python$ export PIP_INDEX_URL=https://gitlab.com/api/v4/pro
jects/riseproject%2Fpython%2Fwheel_builder/packages/pypi/simple
(venv) ubuntu@ubuntu:~/sandbox/python$ time pip install numpy
Looking in indexes: https://gitlab.com/api/v4/projects/riseproject%2Fpython%2Fwheel_build
er/packages/pypi/simple
Collecting numpy
  Downloading https://gitlab.com/api/v4/projects/56254198/packages/pypi/files/118451f2e8f
69584051d71c60a6555ec2be7ae3edd3f9e47e5e40c414a070586/numpy-2.2.2-cp312-cp312-manylinux_2
_35_riscv64.whl (10.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.2/10.2 MB 3.5 MB/s eta 0:00:00

Installing collected packages: numpy
Successfully installed numpy-2.2.2

real    0m21.613s
user    0m17.462s
sys     0m1.163s
(venv) ubuntu@ubuntu:~/sandbox/python$
```

Challenges: Our Experience (So Far)

What Are the Obstacles?

- **General:** (Long build duration) x (multiple Python versions)
- **Lack of distro support** - ideally, use LTS RedHat-based distro
  - RockyLinux in May?
- **Canonical NaN** - Many riscv64 floating point arithmetic instructions return “Canonical NaN”
  - **test failures in packages that test NaNs**, e.g. numpy, torch (possibly others)
- Upstream generally doesn’t test for riscv64
  - Tests don’t necessarily pass **even when wheels build**
  - riscv64 infrastructure isn’t there (**yet**)
- **Responsibility for IP** - e.g. making sure LICENSE files are packaged appropriately

Adding Package Support: Complexity Levels

1. **Easy:** Can just add new version in CI script, trigger build
2. **Medium:** As "Easy", but also requires some tweaks to build environment (e.g. new dependencies pre-installed), custom patches added to mirror of upstream repo
3. **Hard:** "Medium" but requires careful review of test outputs, and/or major overhaul of CI scripts (e.g. change between build backends)

What's Next?

- Targeting **top-30 PyPI packages** + **vllm**, **haystack** dependencies
  - More objectives likely added over time
- Continue maintenance and support as new releases come out for each package
- **Work with upstream** to make riscv64 more accessible
  - **Already started** by reporting issues, discussing
  - Once projects like **manylinux** support riscv64 and **PyPI** accepts riscv64-based wheels, we can shift focus to upstream first
- Discussing with the community - **including you!**

Special Thanks

- RISE Project
- Mark Ryan, Julien Stephan
- The Python and RISC-V communities

Thank You

Questions?