WheelNext Open-Source Initiative

Jonathan Dekhtiar

~

February 2025

01 Problem Statement

What is Wheel-Next - Why fixing Python Packaging ?

Problem 1: PyPI.org and scaling

- Package size limitation (100MB / file by default - hard limit @ 1GB)
- Project size limitation (10 GB / project by default)
- Backlog of "exemption requests": https://github.com/pypi/support
  - Most of the time: taking months to process
  - PyTorch was stuck for months - it took significant effort both NV & META to “unstuck”
  - Exact same with cuDNN and other “accelerated libraries”.
  - Might get slightly better with “PyPI Organization”

What is Wheel-Next - Why fixing Python Packaging ?

Problem 2: Poor “User Experience” using different package indexes.

- External Package Hosting seems to be “just difficult” and difficult to get consensus on.
- No mechanism ~ standard across installers ~ to express relative index priority.
- Opportunity for package name collision across different indexes => Security concerns.

What is Wheel-Next - Why fixing Python Packaging ?

Problem 3: Dealing with “compiled binaries/libs” is difficult.

- No support for symlinks in Python wheels - critical to reduce “packaging bloat” https://pypackaging-native.github.io/other_issues/#lack-of-support-for-symlinks-in-wheels

- Lack of safe, robust, and portable approach for sharing native libraries between wheels
  - Duplication of common Dynamic Shared Objects (DSOs)
  - Increase “package bloat” and load on PyPI infrastructure.

What is Wheel-Next - Why fixing Python Packaging ?

Problem 4: Optional-Dependencies lack the ability to express a “default”

- Optional Dependencies aka. “extras” are often used to express different “backends”
  - API Backend: fastAPI vs Flask vs Django DRF
  - GUI Backend: tkinter vs pyqt
  - Compute Backend: CPU vs GPU vs FPGA vs ASIC
  - Etc.

- **Problem:** You need the user to specify at least one backend
  - User must RTM (i.e. “Read the Manual)

- **UX:** Overall, the user experience is just bad because we lack that “simple feature”.
  => Downloads “systematically” dependencies not always required.

What is Wheel-Next - Why fixing Python Packaging ?

Problem 5: Python Packaging lacks the ability to finely describe “hardware”

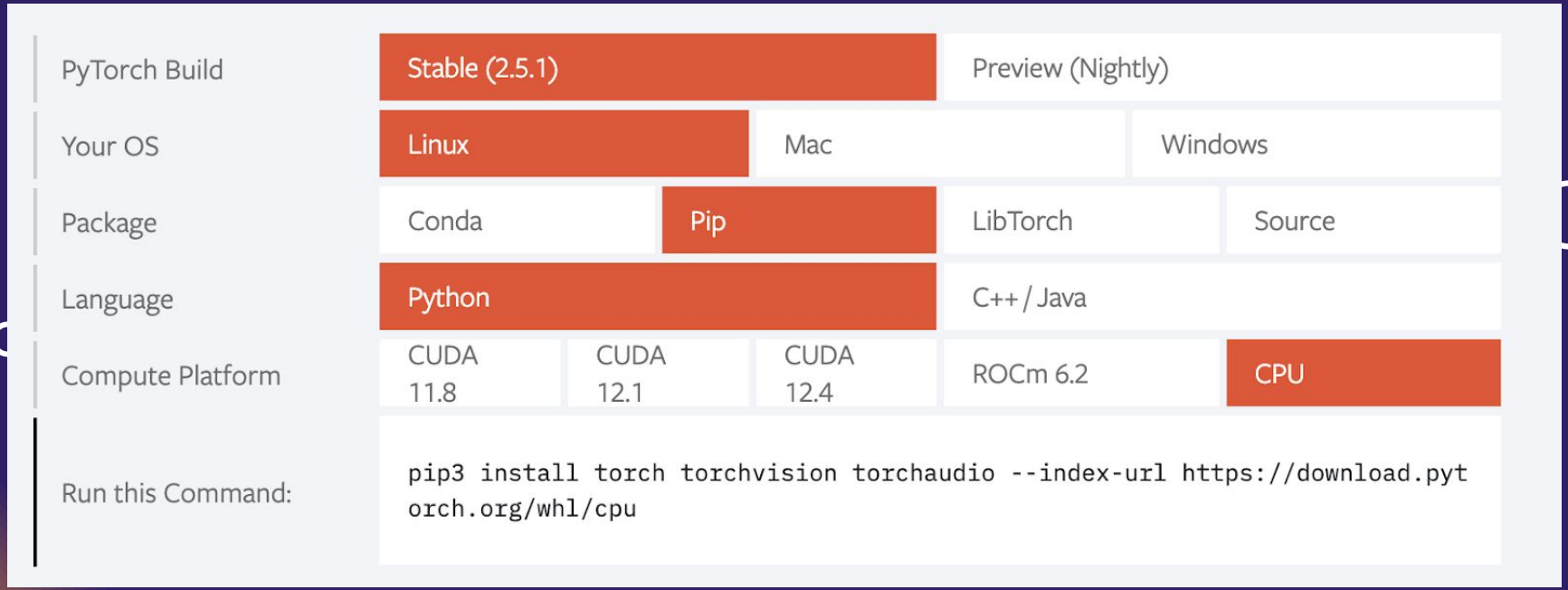
- No way to accurately describe the “hardware platform”
  - What type of accelerators do you have (e.g. CUDA 11, CUDA 12, ROCM, TPU, etc.)
  - What “compute capability” (e.g. SM 90, SM 85, etc.)
  - What ARM version (e.g. ARMv7, ARMv8, etc.)
  - What special CPU instruction (e.g. AVX512)

- What about describing FPGA / ASIC support ?

- What about specific hardware function (e.g. AV1 encoding/decoding) ?

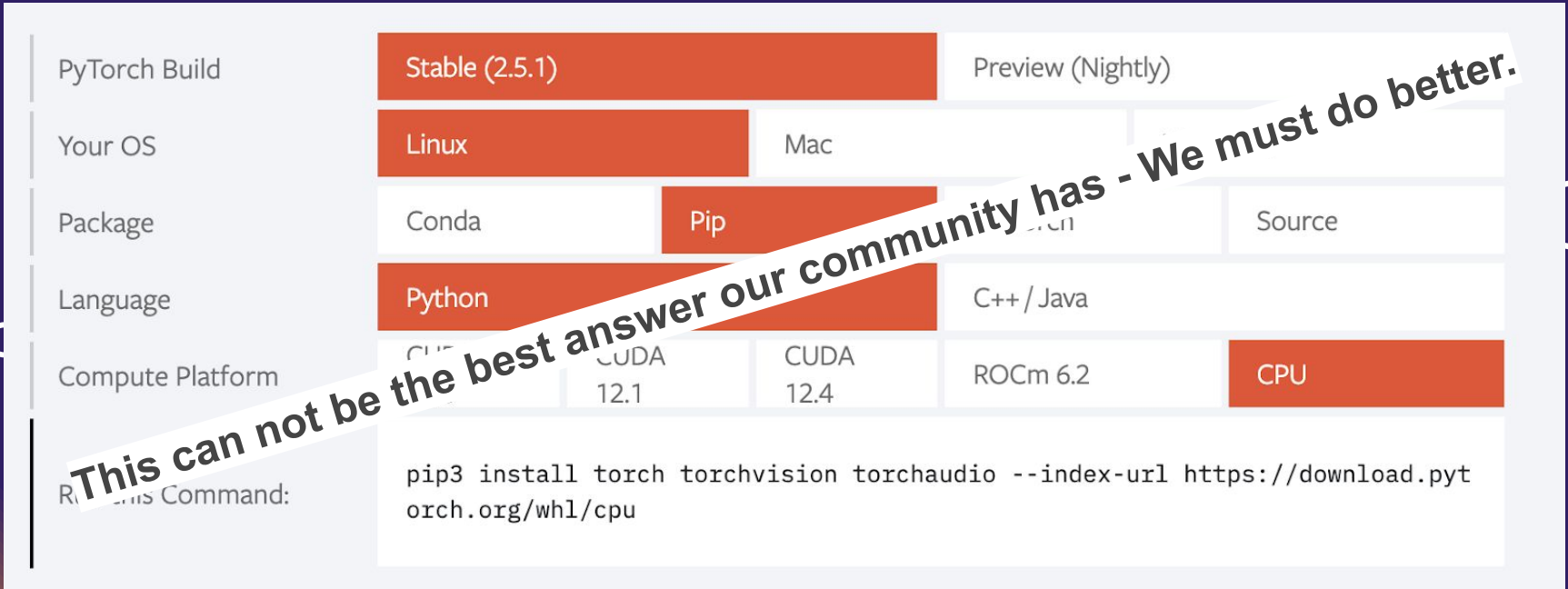
What is Wheel-Next - Why fixing Python Packaging ?

Problem 5: Python Packaging lacks the ability to finely describe “hardware”



What is Wheel-Next - Why fixing Python Packaging ?

Problem 5: Python Packaging lacks the ability to finely describe “hardware”



02 WheelNext Proposals

PEP 771: Default Extra Requires

```
Default-Extra: flask
Provides-Extra: fastapi
Requires-Dist: fastapi; extra == "fastapi"
Provides-Extra: flask
Requires-Dist: flask; extra == "flask"
Provides-Extra: minimal
```

```
[project]
name = "package"
version = "1.0.0"
default-optional-dependency-keys = ["flask"]

[project.optional-dependencies]
fastapi = ["fastapi"]
flask = ["flask"]
```

PEP 771: Default Extra Requires

```
Default-Extra: flask
Provides-Extra: fastapi
Requires-Dist: fastapi; extra == "fastapi"
Provides-Extra: flask
Requires-Dist: flask; extra == "flask"
Provides-Extra: minimal
```

```
pip install package            # install package & extra [flask]
pip install package[flask]     # install package & extra [flask]
pip install package[fastapi]   # install package & extra [fastapi]
pip install package[]          # install package - ONLY
```

```
[project]
name = "package"
version = "1.0.0"
default-optional-dependency-keys = ["flask"]

[project.optional-dependencies]
fastapi = ["fastapi"]
flask = ["flask"]
```

PEP XXX [WIP] Build Isolation Bypass for X

Design / Proposal Space:

- Do not disrupt the current `index` and `extra-index` interface and behavior
- Do not disrupt the current resolution mechanism using `index` and `extra-index`
- Introduce a super-dimension “dependency-group”:
  - *Within the group: resolution as “usual”*
  - *Default Behavior: everything happen in group 0 => no user change*
  - *Can create & order different groups by priority*
  - *Create new flags and APIs (e.g. `pip.conf`) to define dependency-groups: non disruptive*

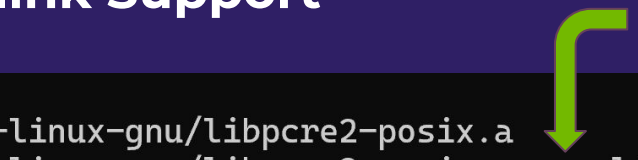
PEP 766: Explicit Priority Choices Among Multiple Indexes

```
1.  [project]
2.  name = "project"
3.  version = "0.1.0"
4.  description = "..."
5.  readme = "README.md"
6.  requires-python = ">=3.12"
7.  dependencies = ["cchardet"]
8.
9.  [tool.uv]
10. no-build-isolation-package = ["cchardet"]
```

Critical to be able to build libraries depending on other package's ABI.

Solution: Let's standardize this "behavior" across build backends & installers.

PEP 778: Symlink Support




```
/usr/lib/x86_64-linux-gnu/libpcre2-posix.a
/usr/lib/x86_64-linux-gnu/libpcre2-posix.so -> libpcre2-posix.so.2.0.3
/usr/lib/x86_64-linux-gnu/libpcre2-posix.so.2 -> libpcre2-posix.so.2.0.3
/usr/lib/x86_64-linux-gnu/libpcre2-posix.so.2.0.3
```

- Standard Linux idiom of library name symlinking is currently inexpressible
- PEP 778 proposes a mechanism to record and reproduce symlink structure in Python package tooling to allow for shared libraries to be used at runtime and link time, without duplication of bytes on disk or in wheels

WHY: https://pypackaging-native.github.io/other_issues/#lack-of-support-for-symlinks-in-wheels

PEP XXX [WIP] Wheel-Variant

Design / Proposal Space:

- Needs to use a “plugin” to put the responsibility on the “provider side”.
  Why: Installers (pip/uv/etc) can not contain “arbitrary logic” without becoming a mess.

- Needs to allow “arbitrary metadata”
  Why: we can’t know today the use cases of tomorrow.

- Needs to not disrupt the current ways wheels are being built & installed.
  Why: What works today - needs to work in the same fashion - no change.

https://github.com/orgs/wheel-next

Join us on GitHub