Proposal: Wheel Variants

March 21, 2025

Jonathan Dekhtiar, NVIDIA

01 Problem Statement

Why “Wheel Variants” ?

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

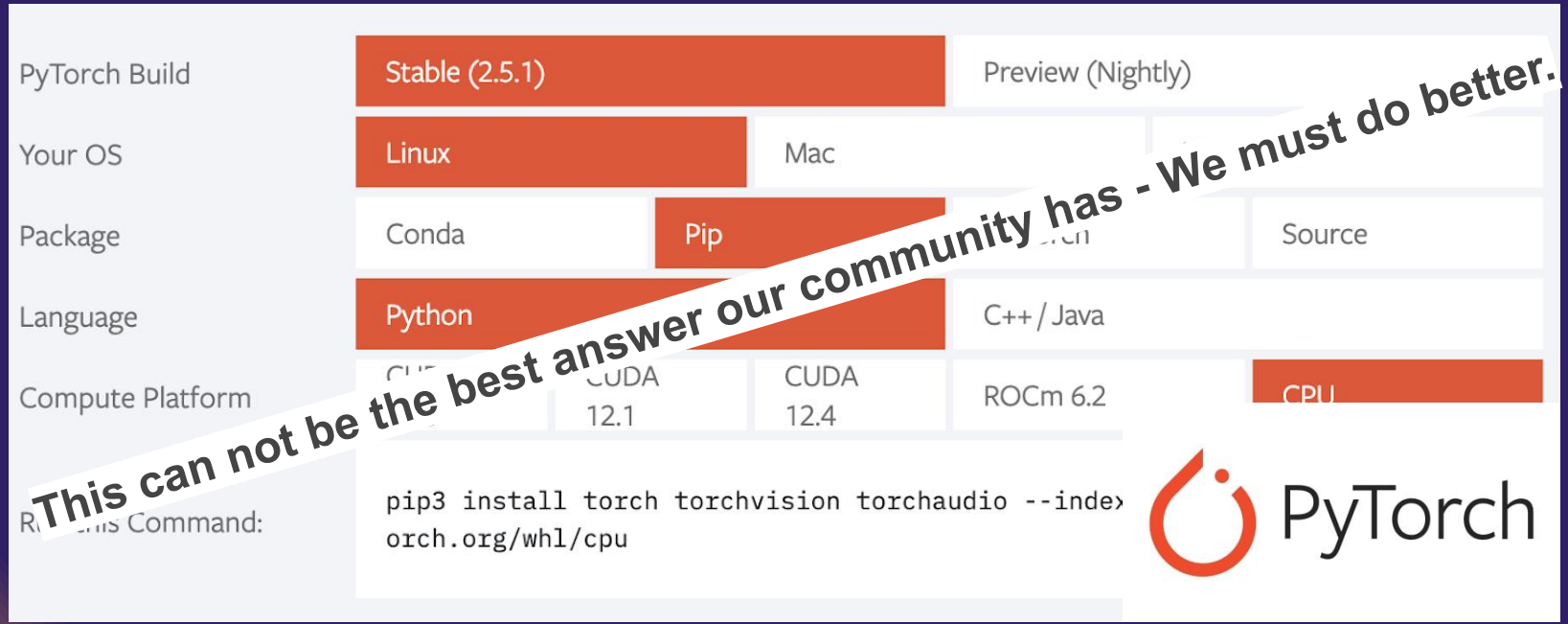
- No way to accurately describe the “hardware platform”
  - What type of accelerators do you have (e.g. CUDA 11, CUDA 12, ROCM, TPU, etc.)
  - What “compute capability” (e.g. SM 90, SM 85, etc.)
  - What ARM version (e.g. ARMv7, ARMv8, etc.)
  - What X86 version (e.g. x86_64-v2, x86_64-v3, etc)
  - What special CPU instruction (e.g. AVX512, SSE, etc.)

- What about describing FPGA / ASIC support ?

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

Why “Wheel Variants” ?

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



Why “Wheel Variants” ?

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

	Linux, x86_64	Linux, aarch64	Mac, x86_64	Mac, aarch64	Windows, x86_64	Windows WSL2, x86_64
CPU	yes	yes	jax≤0.4.38 only	yes	yes	yes
NVIDIA GPU	yes	yes	no	n/a		experimental
Google Cloud TPU	yes	n/a		n/a	n/a	n/a
AMD GPU		no	experimental	n/a	no	no
	n/a	no	n/a	experimental	n/a	n/a
Intel GPU	experimental	n/a	n/a	n/a	no	no

This can not be the best answer our community has - We must do better.

Why “Wheel Variants” ?

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

This can not be the best answer our community has - We must do better.

RAPIDS

Why “Wheel Variants” ?

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

Some References:

- https://pypackaging-native.github.io/key-issues/gpus/
- https://pypackaging-native.github.io/key-issues/simd_support/

02 | User Rationale

Wheel Variant - User Rationale

https://wheelnext.dev/proposals/pepxxx_wheel_variant_support/#rationale

- A user wants to install a version of NumPy that is accelerated for their CPU architecture
- A user wants to install PyTorch / JAX / vLLM that is accelerated for their GPU architecture
- A user wants to install a version of mpi4py that has certain features enabled (e.g. specific MPI implementations for their hardware)
- SciPy wants to provide packages built against different BLAS libraries, like OpenBLAS and Accelerate on macOS. This is something they indirectly do today using different macOS platform tags
- Manylinux cannot express x86-64-v2 requirements in Manylinux_2_34

03 Design & Feature Space

Design Requirement - “Arbitrary Variant Definition”

- We need: Needs to allow “arbitrary metadata”
  => (not GPU, CPU, TPU, FPGA, ASIC etc. or even hardware-focused)

- We do not want: not a “pre-approved list of tags” (e.g. CPU: arm64, x86_64, etc.)

- Why:
  - *We can’t know today the use cases of tomorrow (python for quantum compute?)*
  - *The compute landscape is becoming more complex, more optimized everyday.*
  - *We cannot hope to maintain a list of tags [too many, too many sources]*
  - *Different python communities might use this feature for different purposes*

Design Requirement - “Arbitrary combination of METADATA”

- We need: We need to be able to combine variant information coming from different sources [e.g. GPU Driver version & CPU support for AVX]

- We do not want: Wheel Variants to only be able to include WV information from one source.

- Why:
  - *Wheel Variant “plugins” should be able to “simultaneously describe” a .whl file.*
  - *We need to be able to combine information from different sources [GPU, CPU, etc.]*

Design Requirement - “If you don’t need, you shouldn’t care”

- We need: Wheel Variants should not interfere with the normal “python packaging/installer” workflow & ecosystem.

- We do not want: Wheel Variants to impact packages that don’t need it.

- Why:
  - *This is a niche feature that only affect a small percentages of project*
  - *Not every Python users/maintainers should have to care*

Design Requirement - “Do not break old installers”

- We need: Wheel Variant design should include a mechanism to ensure these “special wheels” will be ignored by installers (e.g. uv, pip) that don’t support them:
  - *Not yet implement*
  - *Old release who didn’t support them*

- We do not want: To confuse an installer that doesn’t support Wheel Variants.

- Why:
  - *It will be very hard to get the PEP accepted if it breaks any previous release of every installers: uv, pip, etc.*

Design Requirement - “No Public API inside PIP”

- We need: We need a standardized “plugin API” that all “build-backends” [setuptools], “installers” [pip, uv], “workflow managers” [pdm, poetry, uv] can use and rely on.

- We do not want: To depend on a public API inside of PIP: *`from pip import XYZ`*

- Why:
  - *To guarantee “tool agnosticism”, we can not depend on a public API in one tool.*
  - *PyPA has consistently refused to maintain any “public user code-API” inside PIP.*

Design Requirement - “Externally Defined: Plugin centric”

- We need: Ability to define “arbitrary metadata/tag” from outside the standard packaging tooling ecosystem (installers, build backends, etc.)

- We do not want: Have to send PRs to any number projects to “declare” the existence of a new metadata / tag.

- Why:
  - *Maintainers of the installer/packaging ecosystem can not be expected to become expert on hardware (CPUs, GPUs, TPUs, ASIC, FPGAs, etc.)*
    => they can’t be expected to review “FPGA-related code”
  - *The maintenance load to review all these PRs would be significant*

Design Requirement - “2D Prioritization: plugin & feature”

- We need:
  - We need a way for users to specify:
    - pluginA > pluginB (e.g. I care more about my GPU support than AVX support)
  - Plugins needs a way to specify:
    - featureA > featureB (e.g. x86-64-v2 is more important than AVX support)

- We do not want: a flat list of plugins and features with no relative priorities

- Why:
  - *Not all features have the same relative importance*
  - *Multiple variants can match a given system (e.g. a generic and a specific)*

Design Requirement - “Scaling should be cheap”

- We need: It shouldn’t matter how many different variants are possible or exists. Deciding which Variant to install should be near instant.

- We do not want: As we scale the number of variant / metadata, the install command take significant time.

- Why:
  - *The search space can become very large very fast*
  - *Combinatorial Products of features*

Design Requirement - “Caching is important or critical”

- We need: A way to cache, manage cache, void cache of the “platform detection and variant resolution”.

- We do not want: Want to re-analyze the entire platform for every single `pip install package` command

- Why:
  - *Loading a bunch of libraries to check versions can be expensive*
  - *System calls to detect X, Y, Z can also be expensive*

Design Requirement - “Forced variant installation”

- We need: A way for an “expert user” to specify: *they desire a specific variant or set of variants in this specific order. Don’t do perform automatic resolution. `[uv] pip --variant=ABC package`*

- We do not want: Have no way for the user to overwrite the automatic resolution if they so wishes.

- Why:
  - *CI Systems may use this*
  - *Advanced users with specific use-cases*
  - *Going around a bug in a specific variant*

Design Requirement - “Forced variant deactivation”

- We need: A way for a user to “disable variant behavior”:
  `[uv] pip install –no-variant package`

- We do not want: Have no way for the user to disable variant installation.

- Why:
  - *CI Systems may use this*
  - *Advanced users with specific use-cases*
  - *Going around a bug in a specific variant*

04 Technical Proposal

Design Requirement - “Arbitrary Variant Definition”

```
# Wheel Variant: dummy_project-0.0.1-py3-none-any-36266d4d+HAL9000.whl

# METADATA File

Variant-hash: 36266d4d
Variant: fictional_hw : architecture :: HAL9000
Variant: fictional_hw : compute_accuracy :: 0
Variant: fictional_hw : compute_capability :: 6
Variant: fictional_hw : humor :: 2
```

- Plugin Name: `fictional_hw`

Design Requirement - “Arbitrary Variant Definition”

```
# Wheel Variant: dummy_project-0.0.1-py3-none-any-36266d4d+HAL9000.whl
# METADATA File
Variant-hash: 36266d4d
Variant: fictional_hw :: architecture :: HAL9000
Variant: fictional_hw :: compute_accuracy :: 0
Variant: fictional_hw :: compute_capability :: 6
Variant: fictional_hw :: humor :: 2
```

- Plugin Name: `fictional_hw`
- Defines “4 variables”

Design Requirement - “Arbitrary Variant Definition”

```
# Wheel Variant: dummy_project-0.0.1-py3-none-any-36266d4d+HAL9000.whl
# METADATA File
Variant-hash: 36266d4d
Variant: fictional_hw :: architecture :: HAL9000
Variant: fictional_hw :: compute_accuracy :: 0
Variant: fictional_hw :: compute_capability :: 6
Variant: fictional_hw :: humor :: 2
```

- Plugin Name: `fictional_hw`
- Defines “4 variables”
- With “1 value assigned per variable”

Design Requirement - “Arbitrary combination of METADATA”

```
# Wheel Variant: dummy_project-0.0.1-py3-none-any-6b4c8391+deepthought+quantum_foam.whl

# METADATA File

Variant-hash: 6b4c8391

Variant: fictional_hw :: architecture :: deepthought
Variant: fictional_hw :: compute_accuracy :: 10
Variant: fictional_hw :: compute_capability :: 10
Variant: fictional_hw :: humor :: 0

Variant: fictional_tech :: quantum :: foam
```

- Legal to combine “metadata” from different sources/plugin.

  => Example: CUDA 12 with AVX512

- Can really be anything so long it follows the “standard format” *<provider_name> :: <variable> :: <value>*

Design Requirement - “If you don’t need, you shouldn’t care”

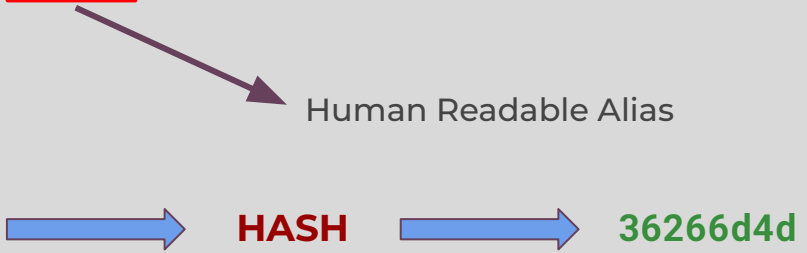
Design Requirement - “Do not break old installers”

```
# Wheel Variant: dummy_project-0.0.1-none-any-36266d4d+HAL9000.whl
# METADATA File
Variant-hash: 36266d4d
Variant: fictional_hw :: architecture :: HAL9000
Variant: fictional_hw :: compute_accuracy :: 0
Variant: fictional_hw :: compute_capability :: 6
Variant: fictional_hw :: humor :: 2
```

Design Requirement - “If you don’t need, you shouldn’t care”

Design Requirement - “Do not break old installers”

```
# Wheel Variant: dummy_project-0.0.1-none-any-36266d4d-HAL9000.whl
# METADATA File
Variant-hash: 36266d4d
Variant: fictional_hw :: architecture :: HAL9000
Variant: fictional_hw :: compute_accuracy :: 0
Variant: fictional_hw :: compute_capability :: 6
Variant: fictional_hw :: humor :: 2
```



Design Requirement - “If you don’t need, you shouldn’t care”

Design Requirement - “Do not break old installers”

```
-rw-r--r-- 1 user user 1266 Feb 20 06:50 dummy_project-0.0.1-py3-none-any.whl

-rw-r--r-- 1 user user 1778 Feb 20 06:50 dummy_project-0.0.1-py3-none-any-36266d4d+hal9000.whl
-rw-r--r-- 1 user user 1773 Feb 20 06:50 dummy_project-0.0.1-py3-none-any-4f8ae729.whl
-rw-r--r-- 1 user user 1777 Feb 20 06:50 dummy_project-0.0.1-py3-none-any-57768a46.whl
-rw-r--r-- 1 user user 1795 Feb 20 06:50 dummy_project-0.0.1-py3-none-any-6b4c8391+deepthought.whl
-rw-r--r-- 1 user user 1779 Feb 20 06:50 dummy_project-0.0.1-py3-none-any-9091cdc4.whl
-rw-r--r-- 1 user user 1760 Feb 20 06:50 dummy_project-0.0.1-py3-none-any-e684be6f.whl
```

```
# METADATA File
Variant-hash: 36266d4d
```

Design Requirement - “Scaling should be cheap”

```
[D 2025-02-20 15:33:01.863 pip.commands.install:108 v0.1.0] [Variant: 0000] `109a2da5`: NOT FOUND ...
[D 2025-02-20 15:33:01.863 pip.commands.install:108 v0.1.0] [Variant: 0001] `c0111c07`: NOT FOUND ...
[D 2025-02-20 15:33:01.863 pip.commands.install:108 v0.1.0] [Variant: 0002] `b5789fbd`: NOT FOUND ...

[...]

[D 2025-02-20 15:33:02.065 pip.commands.install:108 v0.1.0] [Variant: 5984] `8a11085e`: NOT FOUND ...
[D 2025-02-20 15:33:02.065 pip.commands.install:108 v0.1.0] [Variant: 5985] `d0dff1f7`: NOT FOUND ...
[D 2025-02-20 15:33:02.065 pip.commands.install:108 v0.1.0] [Variant: 5986] `44da9896`: NOT FOUND …

[I 2025-02-20 15:33:02.065 pip.commands.install:102 v0.1.0] ###### Best Variant: `9091cdc4` ######
[I 2025-02-20 15:33:02.065 pip.commands.install:104 v0.1.0] Variant-Data: fictional_tech :: quantum :: SUPERPOSITION
[I 2025-02-20 15:33:02.065 pip.commands.install:104 v0.1.0] Variant-Data: fictional_tech :: risk_exposure :: 25
[I 2025-02-20 15:33:02.065 pip.commands.install:104 v0.1.0] Variant-Data: fictional_tech :: technology :: auto_chef
[I 2025-02-20 15:33:02.065 pip.commands.install:105 v0.1.0] ########################################

[I 2025-02-20 15:33:02.065 pip.commands.install:130 v0.1.0] Installing: sandbox_project-0.0.1-py3-none-any-9091cdc4+autochef.whl ...
```

Thank you for your attention