WheelNext Engineering Review

February 20, 2025

Community Updates

Presented by Andy R. Terrel (NumFOCUS / NVIDIA)

🚀 OSS Community Attendee 🚀

PyTorch

JAX

NumPy

astropy

scikit learn

RAPIDS

Numba

scikit-build

CONDA

dmlc XGBoost

NUMFOCUS
OPEN CODE = BETTER SCIENCE

🚀 Companies Attending 🚀

NVIDIA

Meta

Quansight
YOUR DATA EXPERTS

Red Hat

Google

OpenAI

AMD

:probabl.

ASTRAL

ANACONDA

https://github.com/wheelnext

wheelnext

Type / to search

Overview | Repositories 41 | Discussions | Projects 2 | Packages | Teams 10 | People 19 | Insights | Settings

wheelnext

Follow

README.md

WheelNext

LICENSE APACHE-2.0 | DISCORD PYPA | WHEELNEXT | WHEELNEXT.DEV

What is WheelNext

This repo is a place to collaborate on proofs of concept for python wheels in a public open space. It is meant to organize work

View as: Public

You are viewing the README and pinned repositories as a public user.

Get started with tasks that most successful organizations complete.

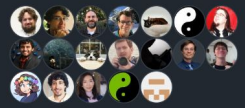
Top discussions this past month

[WheelNext Community] Welcome Everybody !

0 0

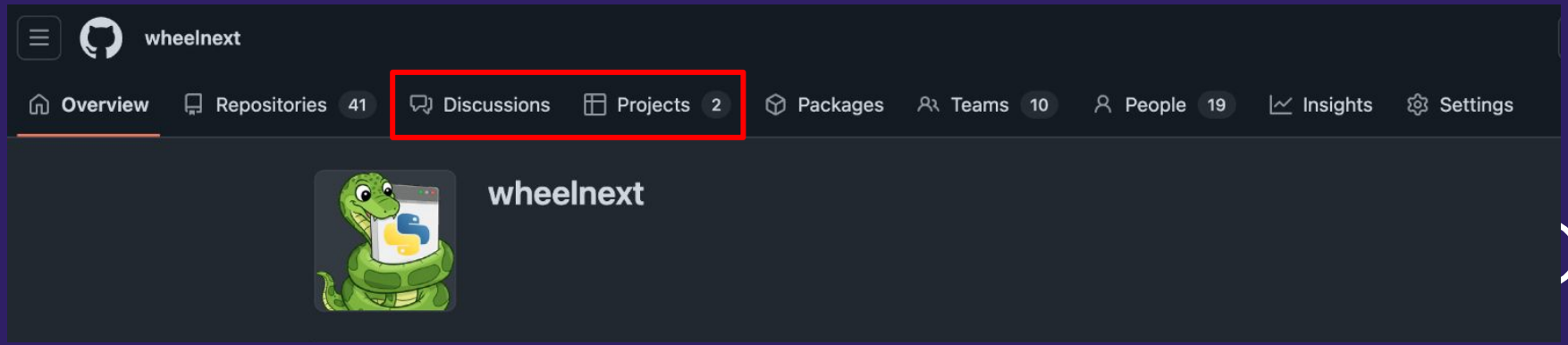
View all discussions

People



Invite someone

https://github.com/wheelnext



https://github.com/wheelnext

faster-cpython / Projects / Fancy CPython Board

Type / to search

Fancy CPython Board

Board | List

Filter by keyword or by field

Todo 12

- Draft: Merge all `BINARY_OP` specializations.
- ideas #441: Identify flow of references in C
- Draft: assist with immortal objects implementation
- multi-core-python #79: PEP for Per-Interpreter GIL
- multi-core-python #80: Move the global runtime config to _PyRuntimeState
- Draft: solve GILState-for-subinterpreters
- ideas #163: Windows startup time
- ideas #102

In Progress 2

- Draft: Implement PEP 669
- Draft: Faster integers

In Review 0

Done 228

- cpython #94808: Metabug: Improving C-level coverage
- ideas #480: Measuring the significance of different benchmarking approaches with A/A testing
- pyperformance #197: Add benchmarks for ctypes function call overhead
- ideas #469: Fix line numbers generated by the compiler
- Draft: Lock Around Extension Loading-related Operations
- Draft: PEP 554 re-writes
- cpython #101659: Isolate the Default Object Allocator between Interpreters

Abandoned 5

- ideas #181: move _Py_unicode_st _PyRuntimeState.glo
- ideas #182: move _Py_exc_state. _Py_exc_state.PyExc _PyRuntimeState.glo
- ideas #64: Faster startup -- Expe
- ideas #261: Work out execution s
- ideas #208: Work towards semi-fo

https://pypackaging-native.github.io/

pypackaging-native

Search

GitHub 152 16

pypackaging-native
- Home
- Meta topics
- Key issues
- Other issues
- Background
- References
- Glossary

Home

Introduction

Packaging is an important and time-consuming part of authoring and maintaining Python packages. This is particularly true for projects that are not pure Python but contain code that needs to be compiled, and have to deal with distributing compiled extensions and with build dependencies. Many projects in the PyData ecosystem - which includes scientific computing, data science and ML/AI projects - fall into that category. This site aims to provide an overview of the most important Python packaging issues for such projects, with in-depth explanations and references.

The content on this site is meant to provide insights and good reference material. This will hopefully provide common ground when discussing potential solutions for those problems or design changes in Python packaging as a whole or in individual packaging tools.

The content is divided into "meta topics" and "key issues". Meta topics are mainly descriptions of aspects of Python packaging that are more or less inherent to the whole design of it, and consequences and limitations that follow from that. Key issues are more specific pain points felt by projects with native code. Key issues may also be more tractable to devise solutions or workarounds for.

Table of contents

https://wheelnext.dev

WheelNext

Search

GitHub 0 2

WheelNext
Welcome to WheelNext
Philosophy and Design Space
Community Updates
WheelNext Proposals
Glossary

Welcome to WheelNext

Introduction

Content to be added here.

Contributing

All contributions are very welcome and appreciated! Ways to contribute include:

- Improving existing content on the website: extending or clarifying descriptions, adding relevant references, diagrams, etc.
- Providing feedback on existing content
- Proposing new topics for inclusion on the website, and writing the content for them
- ... and anything else you consider useful!

The content for this website is maintained on GitHub.

Table of contents
Introduction
Contributing
Acknowledgements

PEP XXX: Wheel Variant

01 Problem Statement

Presented by Andy R. Terrel (NumFOCUS / NVIDIA)

Why “Wheel Variants” ?

<u>Problem:</u> 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”

PyTorch Build	Stable (2.5.1)		Preview (Nightly)		
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python		C++ / Java		
Compute Platform	CUDA 11.8	CUDA 12.1	CUDA 12.4	ROCm 6.2	CPU
Run this Command:	`pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu`				

Why “Wheel Variants” ?

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

PyTorch Build	Stable (2.5.1)		Preview (Nightly)		
Your OS	Linux	Mac			
Package	Conda	Pip			Source
Language	Python		C++ / Java		
Compute Platform		CUDA 12.1	CUDA 12.4	ROCm 6.2	CPU
Run this Command:	pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu				

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

Why “Wheel Variants” ?

Problem: Which “flavor” of PyTorch is this command supposed to download ?

```
$ [uv] pip install transformers    # a package from HuggingFace that depends on PyTorch
```

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 Design Axioms

Presented by Barry Warsaw (Python Steering Council / NVIDIA)

WheelNext - Design Axioms

https://wheelnext.dev/philosophy_and_design_space/

WheelNext - Design Axioms

https://wheelnext.dev/philosophy_and_design_space/#evolution-not-revolution

~ Axiom 1 ~

“Evolution Not Revolution”

WheelNext - Design Axioms

https://wheelnext.dev/philosophy_and_design_space/#if-you-dont-care-now-you-wont-care-later

~ Axiom 2 ~

“If you don't care now, you won't care later”

WheelNext - Design Axioms

https://wheelnext.dev/philosophy_and_design_space/#dont-focus-on-a-single-tool-or-service

~ Axiom 3 ~

“Don't focus on a single tool or service”

WheelNext - Design Axioms

https://wheelnext.dev/philosophy_and_design_space/#favor-backward-compatible-changes-whenever-possible

~ Axiom 4 ~

"Favor backward compatible changes whenever possible"

WheelNext - Design Axioms

https://wheelnext.dev/philosophy_and_design_space/#be-intentful-and-explicit-on-what-is-being-broken-and-why

~ Axiom 5 ~

"Be intentful & explicit on what is being broken and why"

WheelNext - Design Axioms

https://wheelnext.dev/philosophy_and_design_space/#complexity-in-the-tooling-rather-than-user-experience

~ Axiom 6 ~

"Complexity in the tooling rather than user experience"

03 Mode Of Operation

Presented by Barry Warsaw (Python Steering Council / NVIDIA)

WheelNext - Mode of Operation

https://wheelnext.dev/philosophy_and_design_space/#proof-of-concept-minimum-viable-product-first-pep-second

"Proof of Concept First - PEP Second"

WheelNext - Mode of Operation

https://wheelnext.dev/philosophy_and_design_space/#avoid-mission-creep

“Avoid mission creep”

04 User Rationale

Presented by Emma Smith (MyPy Core / NVIDIA)

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

Wheel Variant - User Rationale

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

- A library maintainer wants to build their library for wasm32-wasi with and without pthreads support
- A library maintainer wants to build their library for Pyodide on an Emscripten platform with extensions for graphics compiled in
- A library maintainer wants to provide packages of their game library using different graphics backends
- Manylinux cannot express x86-64-v2 requirements in Manylinux_2_34

05 Design & Feature Space

Presented by Jonathan Dekhtiar (NVIDIA)

Design Requirement - “Arbitrary Variant Definition”

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

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

- <u>Why:</u>
  - *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”

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

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

- <u>Why:</u>
  - *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”

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

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

- <u>Why:</u>
  - *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”

- <u>We need:</u> 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*

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

- <u>Why:</u>
  - *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”

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

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

- <u>Why:</u>
  - *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”

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

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

- <u>Why:</u>
  - *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”

- <u>We need:</u>
  - 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)

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

- <u>Why:</u>
  - *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”

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

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

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

Design Requirement - “Caching is important or critical”

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

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

- <u>Why:</u>
  - *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”

- <u>We need:</u> 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`*

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

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

Design Requirement - “Forced variant deactivation”

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

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

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

06 Technical Proposal

Presented by Jonathan Dekhtiar (NVIDIA)

Design Requirement - “Arbitrary Variant Definition”

```
# Wheel Variant: dummy_project-0.0.1~36266d4d-py3-none-any.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-0.0.1~36266d4d-py3-none-any.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-0.0.1~36266d4d-py3-none-any.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-0.0.1~36266d4d-py3-none-any.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 AVX

- 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”

```
# https://github.com/pypa/pip/blob/main/src/pip/_internal/models/wheel.py#L22

wheel_file_regex = r"""^
    (?P<namever>
        (?P<name>[^\s-]+?)
        -(?P<ver>[^\s-]+?)
    )
    (\-(?P<build>\d[^\s-]*))?
    -(?P<pyver>[^\s-]+?)
    -(?P<abi>[^\s-]+?)
    -(?P<plat>\S+)
\.whl$"""


match = wheel_file_regex.match(filename)
if not match:
      raise InvalidWheelFilename(f"{filename} is not a valid wheel filename.")
```

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

Design Requirement - “Do not break old installers”

```
# https://github.com/pypa/pip/blob/main/src/pip/_internal/models/wheel.py#L22

wheel_file_regex = r"""^
    (?P<namever>
        (?P<name>[^\s-]+?)
        -(?P<ver>[^\s-]+?)
    )
    (\-(?P<build>\d[^\s-]*))?
    (~(?P<variant_hash>[0-9a-f]{8}))?
    (?P<pyver>[^\s-]+?)
    -(?P<abi>[^\s-]+?)
    -(?P<plat>\S+)
\.whl$"""


match = wheel_file_regex.match(filename)
if not match:
      raise InvalidWheelFilename(f"{filename} is not a valid wheel filename.")
```

- A new capture group called “variant hash”
- Illegal with the “former wheel filename regex”
- Uses special character “~” to guarantee:
  - a variant_hash can’t match: `build_id`
  - only free special char RFC 3986 compliant
    - No escaping in bash, windows, macOS
    - No escaping in URLs

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

Design Requirement - “Do not break old installers”

```
# Wheel Variant: dummy_project-0.0.1~36266d4d-py3-none-any.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
```

→ **HASH** → **36266d4d**

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
```

Standard Wheel

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

Wheel Variants

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

Design Requirement - “No Public API inside PIP”

Design Requirement - “Externally Defined: Plugin centric”

```
[project.entry-points."variantlib.plugins"]
my_plugin = "my_plugin.plugin:MyVariantPlugin"
```

```
from variantlib.config import ProviderConfig
from my_plugin import __version__

class MyVariantPlugin:
    __provider_name__ = "my_plugin"
    __version__ = __version__

    def run(self) -> ProviderConfig | None:
        """If the plugin is able to determine this platform/machine supports
        custom "attributes/metadata" (defined and known by this plugin):
        => It returns a `ProviderConfig`, otherwise `None` (aka. ignore me).”""
        return ...
```

Design Requirement - “No Public API inside PIP”

Design Requirement - “Externally Defined: Plugin centric”

```python
from importlib.metadata import entry_points
plugins = entry_points().select(group="variantlib.plugins")

for plugin in plugins:
    logger.info(f"Loading plugin: {plugin.name} - v{plugin.dist.version}")

    # Dynamically load the plugin class
    plugin_class = plugin.load()

    # Instantiate the plugin
    plugin_instance = plugin_class()

    # Call the `run` method of the plugin
    ... = plugin_instance.run()

    # do something with the result of the plugins
```

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

```
# pip.conf or variant.toml

[variantlib]
provider_priority = ["fictional_tech", "fictional_hw"]
```

- Per project: `variant.toml` or inside `pyproject.toml`
- [Tool Specific] PIP - directly inside `pip.conf`
- [Tool Specific] UV - directly inside `uv.toml`

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

```python
# variantlib
from attrs import field
from attrs import frozen

@frozen
class KeyConfig:
    key: str = field()
    values: list[str] = field()

# how to use it
KeyConfig(key="driver_version", values=["12.2.6", "12.2", "12"])
```

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

```python
# variantlib
from attrs import field
from attrs import frozen

@frozen
class ProviderConfig:
    provider: str = field()
    configs: list[KeyConfig] = field()

# how to use it
ProviderConfig(
    provider="provider_name",
    configs=[
        KeyConfig(key="attr_nameA", values=["7", "4", "8", "12"]),
        KeyConfig(key="attr_nameB", values=["3", "7", "2", "18", "22"])
    ]
)
```

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

```python
# variantlib
from variantlib.config import ProviderConfig
from my_plugin import __version__

class MyVariantPlugin:
    __provider_name__ = "my_plugin"
    __version__ = __version__

    def run(self) -> ProviderConfig | None:
        return ProviderConfig(
            provider="my_plugin",
            configs=[
                KeyConfig(key="attr_nameA", values=["7", "4", "8", "12"]),
                KeyConfig(key="attr_nameB", values=["3", "7", "2", "18", "22"])
            ]
        )
```

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

```python
# variantlib
from attrs import field
from attrs import frozen

@frozen
class VariantMeta:
    provider: str = field()
    key: str = field()
    value: str = field()

# Using it
VariantMeta(provider="OmniCorp", key="access_key", value="secret_value")
```

```
Variant: OmniCorp :: access_key :: secret_value
```

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

```python
# variantlib
from attrs import field
from attrs import frozen

@frozen
class VariantDescription:
    data: list[VariantMeta] = field()

# how to use it
VariantDescription([
    VariantMeta(provider="gpu_provider", key="driver_version", value="A.B.C"),
    VariantMeta(provider="cpu_provider", key="avx512", value="true"),
])
```

```
Variant: gpu_provider :: driver_version :: A.B.C
Variant: cpu_provider :: avx512 :: true
```

→ **HASH** → **abcd1234**

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

```
config_custom_hw = ProviderConfig(
    provider="custom_hw",
    configs=[
        KeyConfig(key="driver_version", values=["1.3", "1.2", "1.1", "1"]),
        KeyConfig(key="hw_architecture", values=["3.4", "3"]),
    ],
)

config_networking = ProviderConfig(
    provider="networking",
    configs=[
        KeyConfig(key="speed", values=["10GBPS", "1GBPS", "100MBPS"]),
    ],
)
```

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

Ordering / Prioritization logic:

- More metadata match => Better
- Plugin A > Plugin B => User defined
- PluginA.featureA > PluginA.featureB => Plugin defined

Consequence:

- A variant tagged by all plugin (e.g. GPU & CPU variant) is prioritized over “just GPU or just CPU”
- A variant with more “metadata” (e.g. feature1, feature2, feature3, etc.) is more specific => more prioritized

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

Example:

- Plugin A => featureA
- Plugin B => featureB

Order:

- [pluginA.featureA, pluginB.featureB] => hash => abcd1234
- [pluginA.featureA] => hash => 01234567
- [pluginB.featureB] => hash => ab12cd34

Design Requirement - “Scaling should be cheap”

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

[...]

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

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

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

Design Requirement - “Forced variant deactivation”

```
[uv] pip install --no-variant dummy_project
```

Design Requirement - “Forced variant installation”

```
[uv] pip install --variant=9091cdc4 dummy_project
```

Design Requirement - “Caching is important or critical”

```
# First call - analyze the platform
[uv] pip install dummy_project

# Second call - reuse the platform analysis
[uv] pip install sandbox_project
```

07 Parts that needs work

Presented by Jonathan Dekhtiar (NVIDIA)

Variant Build “user experience”: Build Backend

- <u>We need:</u>
  - A build backend that support Wheel Variant to “demo the idea”.
  - What should be standardized between build backends and what should not.

Variant Build “user experience”: Build Matrix

- <u>We need:</u> A smooth experience to build a large matrix of variants
  => Let’s build 200 variants of PyTorch.

- <u>We do not want:</u> A complicated process to do that
  - Packager experience should be simple and intuitive
  - No way to define a cross product of “features”

Validating “plugin” design to work with `uv`

- We need: Plugin to be functional with both pip/poetry/pdm/hatch/uv/etc.

- Potential Problem (to verify):
  - “entrypoint” is a very python-based feature and plugins provide a Python interface. Let’s ensure `uv` can effectively call the python interface (from ruff) and cache the result.

  - If not, we need to find a better idea

Is a “variant hash” the best approach ?

- Pros:
  - It’s incredibly fast => hash table
  - Allows arbitrary combination of any arbitrary metadata
- Cons:
  - I want the CUDA 12 and AVX512 package => which one is it ?
  - No way to have “named configurations”

MVP from `mockpip` to “real” `pip`

- We need: A real end-to-end implementation with pypa/pip
- We do not want: A proof-of-concept using a super minimalist and narrow “mock”pip implementation.

Verifying Scaling => QuanSight RETEX

- <u>We need:</u> Verify this approach scales to crazy size

Writing the PEP

Community Engagement

Presented by Jonathan Dekhtiar (NVIDIA)

WheelNext & Community - OSS Community Engagement

Save-The-Date: Friday March 21st 2025 ~ 9am -> 1.30pm

WheelNext Community Summit @ META [Menlo Park]

- Validating and refining WheelNext's roadmap
- Aligning on proposals & problem statements
- Working together on common solutions for the Accelerated Compute Space

Attendees:

Companies: *Anaconda, Astral.sh, Amazon/AWS, Bloomberg, Google, META, Microsoft, NVIDIA, Quansight, RedHat*

OSS: Astropy, Jupyter, GPU-Mode, Numba, Numpy, Scikit-Learn, XGBoost, PSF (Python OSS)

https://github.com/wheelnext

Contribute

https://wheelnext.dev

Participate

https://discuss.python.org/c/packaging/

Let's engage

discord.com/channels/803025117553754132/

Join us on Discord

Thank you for your attention