Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б15-мм

Автоматизация сборки, тестирования и публикации кроссплатформенного pip-пакета для Desbordante

ЯКШИГУЛОВ Вадим Наилевич

Отчёт по учебной практике

в форме «Решение»

Научный руководитель:
асс. кафедры ИАС Г. А. Чернышев

Санкт-Петербург
2023

Оглавление

Введение

В современном мире очень большое разнообразие платформ, архитектур и операционных систем. Поэтому высокопроизводительные приложения, часто написанные на компилируемых языках программирования, очень тяжело доставлять до конечного пользователя.

Так, Desbordante[1], высокопроизводительный наукоемкий профилировщик данных, написанный на C++, доступен для пользователей в виде компилируемого Shared Object модуля для Python, а также в виде CLI, написанного на C++. Однако каждому пользователю, который захочет воспользоваться любым из этих интерфейсов, придется скомпилировать приложение, а также установить все зависимости.

Однако в Python существуют другие, менее зависимые от платформы форматы для нативных модулей. Их использование позволяет заранее собрать приложение под определенный набор платформ, а затем опубликовать получившийся pip-пакет (package installer for Python) в PyPI (Python Package Index). Публикация на этом ресурсе повысит доступность Desbordante и увеличит охват пользователей, так как это основной инструмент доставки библиотек и фреймворков в экосистеме Python. А автоматизация этого процесса позволит публиковать новые версии приложения быстрее.

Принимая во внимание эти аспекты, целесообразность публикации проекта Desbordante в PyPI очевидна. Таким образом, автору настоящей работы предстояло автоматизировать публикацию новых версий Desbordante в PyPI.

[1] https://github.com/Mstrutov/Desbordante/ (дата обращения: 12 декабря 2023)

1 Постановка задачи

Целью работы является автоматизация сборки, тестирования и публикации кроссплатформенного pip-пакета для Desbordante. Для её выполнения были поставлены следующие задачи:

1. Сделать обзор инструментов для сборки и публикации pip-пакетов.
2. Реализовать совместимую с pip сборку приложения.
3. Внедрить CI/CD пайплайн для сборки, тестирования и публикации pip-пакета в PyPI.

2 Контекст

2.1 Термины и инструменты

Python Bindings — способ связи библиотек или модулей, написанных на языках программирования отличных от Python (чаще на C или C++), с интерпретатором Python. Основное назначение — делегировать выполнение требовательного к вычислительным ресурсам кода более быстрым языкам программирования.

wheel [2] — формат архива, который содержит все необходимые файлы для установки пакета Python, включая библиотечный код, зависимости и метаданные. Заменяет собой устаревший формат egg.

pybind11 [8] — небольшая библиотека, которая предоставляет Python типы для C++ и наоборот, для создания Python Bindings. Создана для уменьшения объема кода по сравнению со стандартными способами связать C++ и Python.

CMake — система автоматизации сборки программного обеспечения, являющаяся «де-факто стандартом для сборки C++ кода».

setuptools [1] — набор инструментов для управления конфигурацией Python проекта и его зависимостями. Широко используемый инструмент, но требует подробного описания процесса сборки в виде setup.py файла.

manylinux [4] – стандарт созданный Python сообществом для обеспечения совместимости большим количеством различных дистрибутивов Linux. Из-за различий в ABI (Application Binary Interface) и установленных пакетах в системе бинарные пакеты Python не всегда могли запускаться.

CPython и PyPy — две наиболее популярные реализации языка Python.

pyproject.toml [5] — формат для описания информации о проекте, зависимостях и особенностях сборки.

cibuildwheel [7] — инструмент для компиляции wheel под различные платформы: Windows, Linux, macOS. Использует auditwheel [3] для «ис-

правления» созданного в Docker-образе manylinux wheel для того, чтобы добавить больше дистрибутивов в поддержку. Главное преимущество инструмента — сборка интегрирована в CI/CD пайплайн, а значит wheels могут быть собраны на каждый коммит, на новый релиз и перед публикацией в PyPI. Альтернативой может выступать компиляция под каждую платформу без этого инструмента, но тогда потребуется использовать виртуальные машины или Docker-контейнеры под каждую версию CPython, PyPy, операционную систему, дистрибутив Linux и архитектуру, что делает поддержку такого решения необоснованно дорогой.

3 Реализация

3.1 Обзор инструментов

Существует множество инструментов для сборки pip-пакетов из компилируемых языков. В Таблице 1 представлены наиболее популярные из них, поддерживающие C++.

Инструмент	ЯП	Система сборки	Звезды GitHub	pybind11
scikit-build	C/C++/Fortran	CMake	447	Да
meson-python	C/C++/Fortran	Meson	99	Нет
enscons	C/C++	SCON	34	Нет
scikit-build-core	C/C++/Fortran	CMake	125	Да

Таблица 1: Сводная информация по наиболее популярным инструментам сборки pip-пакета

Desbordante написан на C++ с использованием CMake в качестве системы сборки. Так же реализована сборка в Shared Object с помощью pybind11. Поэтому среди всех представленных инструментов нам подходят scikit-build и scikit-build-core. При этом scikit-build-core обладает теми же возможностями, что и scikit-build, а так же некоторыми преимуществами: не зависит от setuptools, активно развивается, использует кеширование между сборками. Более подробно различия описаны в документации проекта.[2]

Исходя из требований к инструменту и уровня поддержки сообществом, было решено выбрать scikit-build-core для сборки Desbordante в pip-пакет.

3.2 Требования к поддерживаемым платформам

После обсуждения с командой проекта требований к сборке проекта было выяснено, что целевыми пользователями приложения являют-

[2]Документация scikit-build-core: https://scikit-build-core.readthedocs.io/en/latest/ (дата обращения: 21 декабря 2023)

ся аналитики, ML-инженеры, специалисты в области баз данных, так как им часто приходится сталкиваться с анализом данных для поиска зависимостей. Было выдвинуто предположение, что целевой платформой, которую используют эти пользователи, является Linux. Поэтому было решено поддержать наиболее популярные дистрибутивы в первую очередь. Python сообщество использует образы manylinux для компиляции пакетов из исходного кода под множество дистрибутивов. Так как в проекте используется C++17, то было решено использовать manylinux2014 — версию образа поддерживающую gcc не ниже 10 версии. В этой версии образа поддерживаются следующие дистрибутивы: CentOS 7 rh-python38, CentOS 8 python38, Fedora 32+, Mageia 8+, openSUSE 15.3+, Photon OS 4.0+, Ubuntu 20.04+.

Помимо прочего, с некоторого момента Desbordante не поддерживает компиляцию под Windows, а значит поддержать эту операционную систему не является возможным и в pip-пакете.

Поддержку macOS и архитектур отличных от x86_64 было решено отложить для того, чтобы успеть реализовать все к подаче публикации “Solving Data Quality Problems with Desbordante: a Demo” на конференцию EDBT (Extending Database Technology)[3].

3.3 Pip-совместимая сборка

Для реализации pip-совместимой сборки в pyproject.toml был добавлен backend для сборки — scikit_build_core.build, а также указаны все Python зависимости необходимые для его сборки: scikit-build-core и pybind11. При этом указать все нативные зависимости необходимые для сборки проекта оказалось невозможным, так как формат pyproject.toml не поддерживает эту функциональность. Хотя работа в этом направлении в Python сообществе ведется [6].

В CMakeList.txt был добавлен дополнительный таргет необходимый для корректной работы scikit-build-core, а так же исправлена ошибка при статическом связывании библиотеки для логирования — она не поз-

[3]Официальный сайт конференции: https://www.edbt.org/ (дата обращения: 21 декабря 2023)

воляла установить пакет без прав суперпользователя.

После этого возможным собрать проект локально в pip-пакет, если все нативные зависимости установлены в системе. В каком-то смысле поменялся лишь способ сборки Shared Object модуля, но именно это позволяет улучшать решение дальше.

3.4 CI/CD пайплайн

На GitHub официальной рабочей группы по инструментам используемым для сборки пакетов Python[4] собрано множество инструментов позволяющих собирать пакеты в различных форматах, включая wheel. Наиболее популярным решением сборки wheels по разные платформы является cibuildwheel. Для его использования требуется минимальное количество действий для конфигурирования. Кроме того инструмент поддерживает не только локальную сборку проекта, но и множество CI/CD платформ, включая GitHub Actions.

Для решения задачи автоматизации процесса сборки был описан workflow — последовательность шагов выполняемых в CI — с использованием GitHub Actions. На Рисунке 1 доступна диаграмма, кратко показывающая все его этапы.

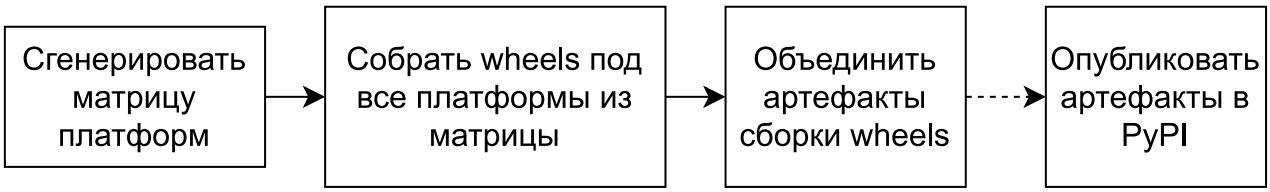

Рис. 1: Диаграмма иллюстрирующая шаги выполнения workflow

Для того чтобы собрать wheels с помощью инструмента под конкретную платформу нужно предварительно подготовить ее платформенный тег — короткое обозначение платформы в соответствии с PEP-513 [3]. Для этого запускается отдельный шаг в workflow по генерации тегов под все реализации (CPython и PyPy) и версии Python, а так же операционные системы. Заметим, что в генерации исключается musllinux —

[4]Ссылка на GitHub аккаунт: https://github.com/pypa (дата обращения: 21 декабря 2023)

альтернативная реализация libc —, так как одна из нативных зависимостей проекта не компилируется при использовании этой реализации.

После этого идет этап сборки wheels с помощью cibuildwheel. На данном этапе все зависимости устанавливаются в Docker-контейнер с образом manylinux2014. А затем проект компилируется из исходного кода. После компиляции и «исправления» с помощью auditwheel wheel тестируется с помощью unit-тестов и сохраняется в артефактах GitHub.

Для ускорения этого этапа каждый wheel с единственным платформенным тегом собирается параллельно остальным. Хотя и приходится компилировать зависимости количество раз сравнимое с количеством платформенных тегов, в конечном счете это ускоряет сборку приложения в десятки раз, так как иначе выполнение было бы последовательным, а количество платформенных тегов оценивается несколькими десятками.

Каждый такой параллельный этап workflow оставляет после себя артефакт. Для удобства wheels полученные после предыдущего этапа объединяются в один артефакт, а единичные экземпляры удаляются.

Последним этапом, запускающимся только при создании нового релиза на GitHub, является публикация в PyPI. Единственный объединенный артефакт оставшийся с предыдущего шага workflow отправляется в PyPI с использованием механизма Trusted Publisher. Этот механизм позволяет создать новый PyPI проект напрямую из CI. Этот подход более безопасный, чем хранение секретного ключа в секретах GitHub, так как генерирует временные токены для доступа к PyPI — это гарантирует, что только workflow с определенного репозитория, конкретного yaml файла имеет разрешение на публикацию новых версий пакета.

Таким образом, была реализована автоматизация сборки, тестирования и публикации в PyPI wheels собранных на основе матрицы платформенных тегов.

3.5 Метаинформация

Для большего информирования потенциальных пользователей приложения была добавлена метаинформация в pyproject.toml — поддерживаемые версии Python, операционные системы, ссылки на репозиторий и сайт проекта, а так же информация о лицензии. Был создан README.md файл эксклюзивно для главной страницы пакета в PyPI, поясняющий нюансы использования и демонстрирующий основную функциональность пакета с помощью примеров.

Заключение

Была автоматизирована сборка, тестирование и публикация pip-пакета для Desbordante.

- Был сделан обзор инструментов для сборки pip-пакетов и выбран подходящий.
- Была реализована совместимая с pip сборка приложения с использованием scikit-build-core.
- Внедрен CI/CD пайплайн для сборки, тестирования и публикации wheels в PyPI для дистрибутивов Linux.

Кроме того, после публикации в PyPI[5] Desbordante был выпущен в релиз с версией 1.0.0. А консольный интерфейс Desbordante был переписан на Python с использованием этого пакета.

За вклад в публикацию релизной версии Desbordante автор настоящей работы был указан в качестве соавтора в статье "Solving Data Quality Problems with Desbordante: a Demo" отправленной на конференцию EDBT.

Количество пользователей, скачавших пакет с PyPI, достигло 1255 согласно сайту для сбора статистики с PyPI.[6]

В будущем можно увеличить количество поддерживаемых платформ, например, macOS или Windows и добавить больше архитектур, например, RISC-V.

[5]Страница пакета в PyPI: https://pypi.org/project/desbordante/ (дата обращения: 21 декабря 2023)

[6]https://www.pepy.tech/projects/desbordante (дата обращения: 21 декабря 2023)

Список литературы

[1] Building and Distributing Packages with Setuptools. — URL: https://setuptools.pypa.io/en/latest/setuptools.html (дата обращения: 12 декабря 2023 г.).

[2] PEP 491 — The Wheel Binary Package Format 1.9. — URL: https://peps.python.org/pep-0491/ (дата обращения: 12 декабря 2023 г.).

[3] PEP 513 — A Platform Tag for Portable Linux Built Distributions. — URL: https://peps.python.org/pep-0513/ (дата обращения: 12 декабря 2023 г.).

[4] PEP 599 — The manylinux2014 Platform Tag. — URL: https://peps.python.org/pep-0599/ (дата обращения: 12 декабря 2023 г.).

[5] PEP 621 — Storing project metadata in pyproject.toml. — URL: https://peps.python.org/pep-0621/ (дата обращения: 12 декабря 2023 г.).

[6] PEP 725 — Specifying external dependencies in pyproject.toml. — URL: https://peps.python.org/pep-0725/ (дата обращения: 12 декабря 2023 г.).

[7] cibuildwheel Documentation. — URL: https://cibuildwheel.readthedocs.io/en/stable/ (дата обращения: 12 декабря 2023 г.).

[8] pybind Documentation. — URL: https://pybind11.readthedocs.io/en/latest/ (дата обращения: 12 декабря 2023 г.).