Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б15-мм

Реализация MVP нового веб-сервера Desbordante

ЯКШИГУЛОВ Вадим Наилевич

Отчёт по учебной практике
в форме «Решение»

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

Санкт-Петербург
2024

Оглавление

Введение

Развитие информационных систем со временем неизбежно приводит к их усложнению. Изначально простые и понятные решения, созданные для выполнения конкретных задач, с добавлением новых возможностей становится дорого и трудно поддерживать.

Так, серверная часть Desbordante[1] — открытого профилировщика данных [3] — органично развивалась той же командой разработчиков, которая писала и фронтенд на JavaScript, и основное приложение на C++. Разработчики выбирали технологии на ходу, ориентируясь на текущие потребности проекта, что позволяло оперативно решать задачи. Однако со временем, по мере роста и усложнения системы, добавление новой функциональности стало требовать знания JavaScript, Python и C++, что затрудняло адаптацию новых разработчиков.

Добавление нового способа взаимодействия с Python в виде pip-пакета[10] позволяет теперь писать бэкенд только на этом языке программирования. А развитые системы для постановки задач, такие как Celery, позволят уменьшить кодовую базу и достичь того же поведения, что и при использовании Kafka [1] и Docker-контейнеров [4].

Принимая во внимание эти аспекты, а так же увеличение количества заинтересованных разработчиков после официального релиза[2], реинжиниринг бэкенда представляется необходимым. Таким образом, задача автора настоящей работы состояла в разработке новой реализации в качестве минимально жизнеспособного продукта (MVP).

[1] https://github.com/Desbordante/desbordante-web (дата обращения: 20 августа 2024)

[2] https://news.ycombinator.com/item?id=40063137 (дата обращения: 20 августа 2024)

1 Постановка задачи

Целью работы является разработка MVP нового веб-сервера Desbordante. Для её выполнения были поставлены следующие задачи:

1. Сделать обзор функциональных компонентов предыдущей реализации.
2. Определить требования к новой системе.
3. Реализовать MVP в соответствии с требованиями.

2 Обзор

2.1 Обзор существующего решения

Полная архитектура веб-приложения описана на Рисунке 1. Можно выделить пять основных компонентов серверной части:

1. Бэкенд на Node.js
   (a) Постановка новых задач по переданной конфигурации
   (b) Получение информации о конкретной поставленной задачи (результат и статус)
   (c) Поддержка авторизации и менеджмента доступа к постановке задач
   (d) Уведомление Kafka Consumer'а о создании новой задачи через Kafka
2. Очередь Kafka [1] для информирования Kafka Consumer'а
3. PostgreSQL [6] — СУБД, хранящая информацию о задачах, их статусах, пользователях и файлах
4. Kafka Consumer, написанный на Python, для оркестрации алгоритмов Desbordante
5. Алгоритмы Desbordante (Desbordante Core) написанные на C++ и изолированные с помощью Docker-контейнеров для предупреждения падения сервера из-за возможных ошибок во время выполнения

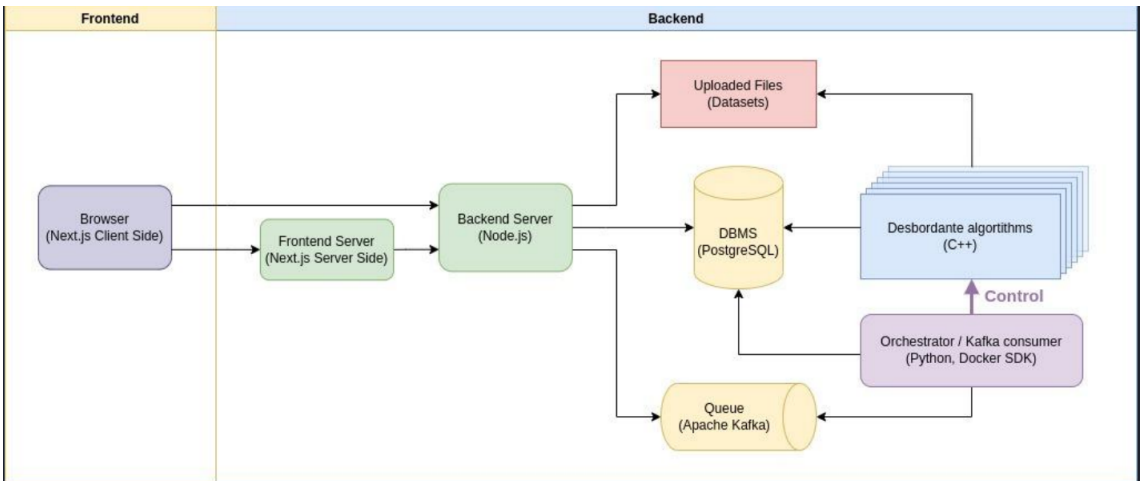

Рис. 1: Архитектура исходного веб-приложения

Обычный сценарий использования веб-приложения со стороны пользователя и веб-сервера выглядит так:

1. Пользователь выполняет вход в веб-интерфейсе или выполняет дальнейшие действия анонимно.

2. Пользователь загружает датасет для дальнейшей обработки.

3. Веб-сервер сохраняет информацию о датасете в базе данных и файл на жестком диске.

4. Пользователь выбирает примитив, алгоритм и задает параметры для его работы на загруженном датасете.

5. Веб-сервер создает задачу в базе данных и кладет идентификатор задачи в очередь Kafka [1]. Затем Kafka Consumer забирает идентификатор из очереди и запускает соответствующий контейнер с Desbordante Core. Алгоритмы после завершения работы записывают в базу данных информацию о статусе задачи и вычисленных зависимостях. Если контейнер упал с ошибкой во время выполнения, то Kafka Consumer запишет в базу данных информацию об ошибке.

6. Веб-интерфейс запрашивает информацию о задаче. Если задача

выполнилась или упала с ошибкой, то пользователю выводится ее статус и найденные зависимости, если нашлись.

2.2 Desbordante Core в виде pip-пакета

В начале 2024 года был опубликован pip-пакет Desbordante[10]. Этот пакет позволяет использовать Desbordante Core напрямую из Python, что упрощает интеграцию с другими Python-приложениями.

2.3 Основные требования к веб-серверу

В ходе обсуждения со стейкхолдерами был выяснен следующий список требований:

1. В случае падения Desbordante Core в результате SEGFAULT, недостатка памяти, чрезмерной загрузки процессора и прочих причин, веб-сервер должен продолжать свою работу и продолжать отвечать на запросы.
2. Конфигурация каждой задачи должна проверяться на стороне веб-сервера до постановки задач в Desbordante Core.
3. Веб-интерфейсу должна быть доступна полная типизированная спецификация API для упрощения интеграции.
4. Новая реализация должна позволить быстрее добавлять новые примитивы.
5. Должен использоваться язык Python
6. API должен быть очень простым

3 Реализация

3.1 Обоснование выбора технологий

Так как среди требований явно указан язык программирования Python, были выбраны следующие ключевые технологии, отличающиеся высокой популярностью:

1. FastAPI [5] — фреймворк для разработки типизированного API с автоматической генерацией документации в формате спецификации OpenAPI [9].
2. Celery [2] — система управления задачами, обеспечивающая распределение их выполнения по потокам (процессам) или компьютерам.
3. SQLAlchemy [8] — наиболее популярная библиотека ORM, поддерживающая работу с PostgreSQL [6].
4. Pydantic [7] — библиотека для описания типизированных структур с автоматической валидацией переданных параметров.

Выбор FastAPI [5] обусловлен не только его растущей популярностью, но и требованиями к автоматической генерации документации. Этот фреймворк возвращает спецификацию OpenAPI [9], которая может быть использована для генерации клиентских библиотек.[3] Pydantic [7], входящий в состав FastAPI [5], позволяет типизировать как API, так и объекты предметной области.

Использование Celery [2] обеспечивает изоляцию задач от основного сервера. В случае, если задача завершается с ошибкой или возвращает статус, отличающийся от нуля, Celery [2] может перехватить эту информацию, что позволяет сохранить её в базе данных. Аналогичные решения, такие как Taskiq, не поддерживают эту функциональность «из коробки»[4].

[3] https://fastapi.tiangolo.com/advanced/generate-clients/ (дата обращения: 28 сентября 2024)
[4] https://github.com/taskiq-python/taskiq/issues/212 (дата обращения: 28 сентября 2024)

А SQLAlchemy [8] позволяет писать запросы к базе данных значительно быстрее, что особенно важно для MVP.

3.2 Определение API

Минимальный API для постановки и получения задач должен предоставлять следующие возможности:

1. Загрузка датасета
2. Создание задачи по заданной конфигурации
3. Получение текущего статуса задачи и ее результата

3.3 Архитектура

На Рисунке 2 представлена архитектура разработанного приложения.

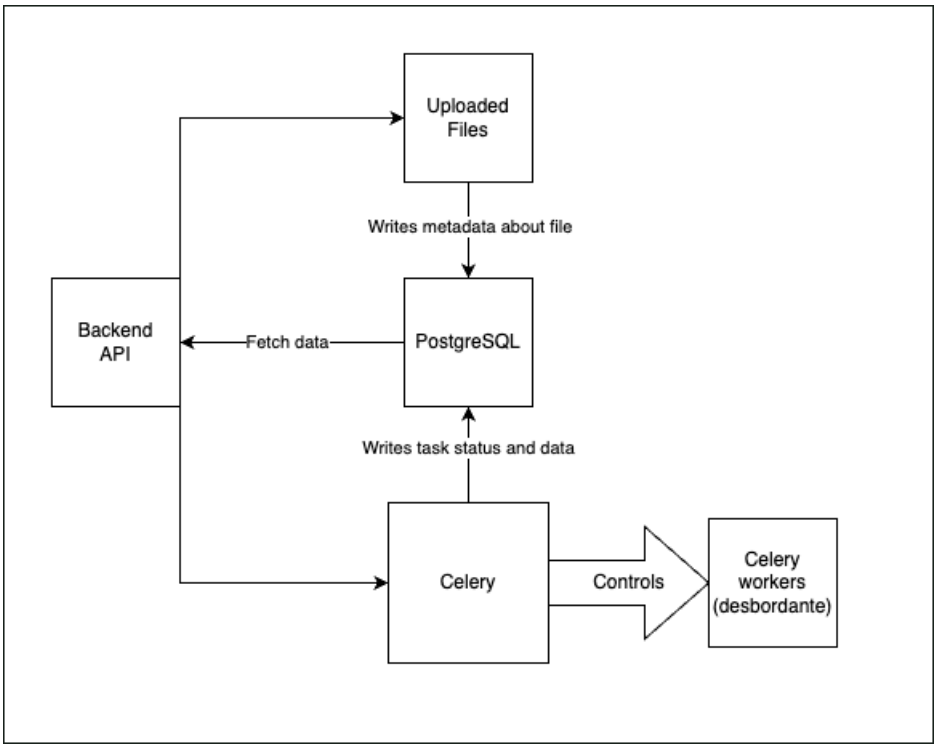

Рис. 2: Архитектура нового веб-приложения

С помощью API пользователи могут создавать датасеты, при этом метаданные хранятся в базе данных.

API обеспечивает возможность получения информации о конкретной задаче по её уникальному идентификатору (ID) из базы данных. Ответ включает детальное описание ошибки, если таковая возникла, а также полную конфигурацию, в которой была выполнена задача, и полученный результат. Сохраняются также название примитива и алгоритма, что позволяет однозначно идентифицировать возвращаемый тип.

Дополнительно API предоставляет функциональность для запуска задачи в воркере Celery [2] на основании переданной конфигурации. По завершении выполнения Celery передает информацию о результате, которая сохраняется в базе данных.

Кроме того, через API осуществляется доступ к задаче и датасету по их уникальным идентификаторам (ID) из базы данных.

4 Эксперимент

4.1 Сравнение с предыдущей системой

Ниже представлена таблица сравнивающая две версии приложения старую и новую.

Количество строк кода подсчитано утилитой cloc, а время добавления, начиная с даты постановки задачи, заканчивая принятым PR.

Сравнение приведено для добавления примитива MFD и единственного доступного алгоритма MetricVerifier.

Параметр	**Старая**	**Новая**
LOC Python	23	267
LOC C++	50	0
LOC Node.js	340	0
Time/PR	3 месяца	2 недели

Таблица 1: Сравнение старой и новой реализаций приложения

4.2 Вывод

Новая реализация значительно уменьшает количество кода, который нужно написать, чтобы добавить новый примитив в веб-сервер, а так же уменьшает время на полное выполнение задачи.

Заключение

В результате работы был разработан MVP нового веб-сервера Desbordante.

1. Проведен обзор функциональных компонентов предыдущей реализации.
2. Определены ключевые требования к новой системе.
3. Реализован MVP в соответствии с требованиями.
    - Реализован сервер на FastAPI, обеспечивающий простоту использования и автоматическую генерацию документации API.
    - Внедрена система управления задачами с использованием Celery для повышения устойчивости и изоляции выполнения алгоритмов.
    - Создан минимальный API для загрузки датасетов, создания задач и получения их статуса.

В будущем можно реализовать модуль для работы с пользователями, а так же улучшить архитектуру существующего решения для большей гибкости.

Список литературы

[1] Apache Kafka. — URL: https://kafka.apache.org/.

[2] Celery - Distributed Task Queue. — URL: https://docs.celeryq.dev/en/stable/.

[3] Chernishev George, Polyntsov Michael, Chizhov Anton et al. Desbordante: from benchmarking suite to high-performance science-intensive data profiler (preprint). — 2023. — 2301.05965.

[4] Docker: Accelerated Container Application Development. — URL: https://www.docker.com/.

[5] FastAPI. — URL: https://fastapi.tiangolo.com/ru/.

[6] PostgreSQL: The World's Most Advanced Open Source Relational Database. — URL: https://www.postgresql.org/.

[7] Pydantic. — URL: https://docs.pydantic.dev/latest/.

[8] The Python SQL Toolkit and Object Relational Mapper. — URL: https://www.sqlalchemy.org/.

[9] The world's most widely used API description standard. — URL: https://www.openapis.org/.

[10] Якшигулов Вадим. Автоматизация сборки, тестирования и публикации кроссплатформенного pip-пакета для Desbordante. — 2024. — Отчёт по учебной практике.