Рахимов Юнель Рамилевич
Бакалавр
Российского технологический университет МИРЭА, ИИТ
Мирошник Егор Иванович
Бакалавр
Российский технологический университет МИРЭА, ИИТ

РАЗРАБОТКА ПЛАТФОРМЫ РАСПРЕДЕЛЁННОЙ ТРАНСКРИБАЦИИ АУДИОДАННЫХ ДЛЯ ОБРАБОТКИ НОВОСТНЫХ ЭФИРОВ

Аннотация: В статье рассматривается задача автоматизации транскрибации больших объёмов аудиоданных на примере новостных эфиров. Показано, что отдельные модели автоматического распознавания речи не обеспечивают полноценную инфраструктуру для потоковой и пакетной обработки медиаконтента. Предложена архитектура программной платформы, объединяющей модули приёма данных, предобработки, транскрибации, управления задачами и хранения результатов. Реализация выполнена на базе Python, FastAPI, PostgreSQL, RabbitMQ, MinIO и модели T-one. Платформа поддерживает асинхронную обработку архивных файлов, транскрибацию интернет-потоков и распознавание речи в реальном времени через WebSocket. Результаты тестирования подтвердили работоспособность всех основных сценариев и соответствие заявленным показателям производительности.

Ключевые слова: транскрибация аудиоданных, распознавание речи, распределённая система, потоковая обработка, новостные эфиры, FastAPI, RabbitMQ.

Keywords: audio transcription, speech recognition, distributed system, stream processing, news broadcasts, FastAPI, RabbitMQ.

Введение

Эффективная работа с большими массивами аудиоданных в медиасфере и аналитике требует не только хранения записей, но и их быстрой расшифровки в текстовый формат. В новостных редакциях и исследовательских подразделениях ежедневно формируются записи эфиров, интервью и репортажей, однако ручная транскрибация остаётся трудоёмкой и замедляет подготовку аналитических материалов.

Развитие технологий автоматического распознавания речи (Automatic Speech Recognition, ASR) позволяет существенно ускорить обработку аудиоконтента, однако на практике возникает ряд ограничений. Качество транскрибации зависит от уровня шума, формата записи и числа говорящих [1]. При обработке больших объёмов данных необходимы унификация форматов, распределение вычислительной нагрузки и организация устойчивого хранения результатов [2]. Особую актуальность задача приобретает в сценариях, где одновременно требуется поддерживать обработку архивных файлов, интернет-трансляций и аудиопотока в режиме реального времени.

Целью работы является разработка платформы распределённой транскрибации

аудиоданных, обеспечивающей автоматизацию приёма, обработки, хранения и выдачи результатов транскрибации на примере новостных эфиров.

Анализ предметной области и существующих решений

Основными источниками аудиоданных в рассматриваемой предметной области являются архивные файлы (WAV, MP3, M4A), потоковые интернет-трансляции новостных каналов и сигнал с микрофона пользователя [3]. Практическим примером потокового источника служит интернет-радио «Вести FM», распространяемое через сервер Icecast по протоколу HTTP [4]. Поток передаётся в формате MP3 с битрейтом 192 кбит/с, частотой дискретизации 48 кГц и стереозвуком, тогда как модели ASR, как правило, требуют моносигнал с пониженной частотой дискретизации [5].

Перед транскрибацией выполняется предобработка: конвертация формата, преобразование стерео в моно, ресемплинг и сегментация длинных записей на фрагменты фиксированной длительности [6]. Сегментация облегчает потоковую обработку и распределение нагрузки между вычислительными узлами.

Для сравнения существующих ASR-решений были выбраны OpenAI Whisper, t-tech/T-one и Vosk. Критерии оценки включали производительность [7], точность распознавания, функциональность [8], масштабируемость и возможность интеграции в автоматизированные контуры обработки.

Whisper обеспечивает высокую точность на шумных записях и поддерживает множество языков, но требует значительных вычислительных ресурсов при обработке больших объёмов данных [9]. Модель T-one ориентирована на потоковое распознавание русской речи, обрабатывает аудио чанками по 300 мс и демонстрирует высокое качество на медиапотоках при относительно небольшом размере (~70 млн параметров) [10]. Vosk подходит для офлайн-интеграции, однако уступает современным нейросетевым моделям по качеству на сложных записях. Сводная оценка представлена в Таблице 1.

Таблица 1. **Сравнительный анализ ASR-моделей**

Критерий	Whisper	T-one	Vosk
Точность	очень высокая	высокая для русской речи	средняя
Потоковый режим	ограниченно	да	да
Масштабируемость	при наличии GPU	да	да
Инфраструктура обработки	нет	нет	нет

Проведённый анализ показал: эффективные ASR-модели решают задачу распознавания, но не предоставляют средств для автоматизации приёма потоков, постановки задач в очередь, хранения результатов и интеграции с внешними системами [11]. Это обосновывает необходимость специализированной платформы, объединяющей транскрибацию, распределённую обработку и управление данными.

Архитектура и выбор технологий

Платформа реализована по модульной клиент-серверной архитектуре [12] и включает следующие компоненты:

- модуль приёма аудиоданных;
- модуль предобработки;
- модуль транскрибации (ASR);
- модуль управления задачами;
- модуль хранения данных;
- веб-интерфейс и программный API.

Модуль приёма обеспечивает загрузку файлов, регистрацию URL потоковых источников и приём сигнала с микрофона. Модуль предобработки приводит аудиосигнал к единому формату, необходимому для работы модели. Модуль транскрибации преобразует подготовленный сигнал в текст. Модуль управления задачами распределяет нагрузку между worker-процессами и обеспечивает асинхронную обработку. Модуль хранения сохраняет транскрипции, метаданные и текстовые чанки, формируемые в потоковом режиме [13].

Математически задача распознавания формулируется как поиск наиболее вероятной последовательности слов W для заданного аудиосигнала X:

$$W^* = \arg max\, P(W|X).$$

В современных системах вероятность $P(W|X)$ оценивается нейросетевыми моделями, обученными на больших речевых корпусах [14].

Общая последовательность обработки включает этапы: получение аудиосигнала, предобработка, сегментация, распознавание, сохранение результатов. При потоковой обработке данные проходят те же этапы, но без ожидания завершения всей записи.

Для реализации выбран язык Python как наиболее подходящий для интеграции с

библиотеками машинного обучения и обработки аудио [15]. Серверная часть построена на FastAPI, для хранения метаданных используется PostgreSQL, для очередей задач — RabbitMQ, а для аудиофайлов — S3-совместимое хранилище MinIO. В качестве ASR-модели выбрана T-one как оптимальная для русскоязычных потоковых сценариев. Развёртывание выполняется в Docker, что обеспечивает переносимость решения [16, 17]. Для ускорения параллельной обработки возможно использование GPU [18].

Состав программного стека приведён в Таблице 2.

Таблица 2. **Программный стек платформы**

Компонент	Технология
Язык и API	Python 3.11+, FastAPI
СУБД	PostgreSQL
Очередь задач	RabbitMQ
Объектное хранилище	MinIO (S3)
ASR-модель	T-tech/T-one
Контейнеризация	Docker

Реализация и сценарии обработки

Платформа поддерживает три режима работы: пакетную обработку архивных файлов, транскрибацию потоковых URL-источников и распознавание речи с микрофона в реальном времени.

При загрузке аудиофайла API-сервер выполняет валидацию, сохраняет файл в MinIO и регистрирует задачу в PostgreSQL. Задача публикуется в очередь RabbitMQ, откуда её получает ASR Worker. Такой подход отделяет приём пользовательского запроса от длительной операции распознавания и обеспечивает асинхронный режим обработки.

Worker загружает аудио из хранилища, выполняет предобработку (декодирование, нормализация каналов и частоты дискретизации), передаёт данные в модуль транскрибации и сохраняет результат в базу данных. Подтверждение сообщения в очереди выполняется только после успешного завершения всех операций, что снижает риск потери задач.

Для потоковой транскрибации с микрофона и URL-источников реализовано взаимодействие через WebSocket. Аудиопоток передаётся фрагментами; при накоплении заданного объёма данных формируется очередной чанк для распознавания. Результаты сохраняются в базе данных не целиком, а в виде текстовых фрагментов, что позволяет

отображать промежуточные результаты пользователю по мере поступления сигнала.

Пользовательский интерфейс реализован на HTML, CSS и JavaScript и объединяет на одной странице сценарии записи с микрофона, загрузки файлов, управления потоковыми источниками и просмотра транскрипций. Дополнительно предусмотрен REST API на базе FastAPI для интеграции с внешними аналитическими системами без использования графического интерфейса.

Логическая модель данных включает сущности аудиофайлов, потоковых источников, задач обработки, транскрипций и текстовых чанков. Такая структура обеспечивает связь между источником аудио, процессом обработки и результатами на всех этапах жизненного цикла данных.

Распределение функций между модулями представлено в Таблице 3.

Таблица 3. **Назначение функциональных модулей**

Модуль	**Назначение**	**Результат**
Приём данных	файлы, URL, микрофон	зарегистрированный вход
Управление задачами	очередь и статусы	задача в БД и RabbitMQ
Предобработка	нормализация аудио	массив для ASR
Транскрибация	ASR-преобразование	текст или чанки
Хранение	БД и MinIO	данные для просмотра и API
API	HTTP-доступ	транскрипции и статусы

Тестирование и оценка эффективности

Тестирование проводилось по двум направлениям: функциональная проверка сценариев и оценка эксплуатационных характеристик. Использовались метрики задержки (latency) [19], пропускной способности (throughput) [20], доли успешно завершённых задач (task success rate) [21], а также проверка корректности API-ответов.

Было выполнено 10 тестовых сценариев: запуск интерфейса, транскрибация с микрофона, загрузка файла, обработка worker-модулем, добавление потокового источника, сохранение чанков, получение результата через API, просмотр накопленных данных, обработка некорректного формата файла и реакция на недоступность RabbitMQ или MinIO [22]. Все сценарии завершились успешно; при ошибочных входных данных система возвращала контролируемые сообщения без нарушения работы остальных модулей.

Результаты функционального тестирования приведены в Таблице 4.

Таблица 4. **Результаты функционального тестирования**

№	Сценарий	Результат
1	Базовый запуск	успешно
2	Транскрибация с микрофона	успешно
3	Загрузка аудиофайла	успешно
4	Обработка worker-модулем	успешно
5	Потоковый URL-источник	успешно
6	Сохранение чанков в БД	успешно
7	Получение результата через API	успешно
8	Просмотр в интерфейсе	успешно
9	Некорректный формат файла	ошибка обработана
10	Недоступность RabbitMQ/MinIO	ошибка зафиксирована

Показатели производительности представлены в Таблице 5. Фактические значения соответствуют или превосходят заявленные требования.

Таблица 5. **Показатели производительности**

Показатель	Требование	Фактическое значение
Постановка задачи в очередь	Менее 2 с	Менее 1 с
Первый текстовый чанк (микрофон)	Менее 5 с	1–2 с
Обработка файла среднего объёма	Менее 60 с	Менее 5 с
Сохранение результата в БД	Менее 3 с	Менее 2 с

Методика оценки соответствовала общепринятым подходам к тестированию программных систем [23].

Заключение

В работе разработана платформа распределённой транскрибации аудиоданных для

обработки новостных эфиров. Предложенная модульная архитектура объединяет приём данных из различных источников, асинхронную обработку через очередь сообщений, потоковую транскрибацию в реальном времени и централизованное хранение результатов. Реализация на базе FastAPI, PostgreSQL, RabbitMQ, MinIO и модели T-one подтвердила практическую применимость решения.

Тестирование показало корректную работу платформы во всех основных режимах: пакетной обработке файлов, работе с интернет-потоками и транскрибации с микрофона. Полученные результаты позволяют рекомендовать платформу для автоматизации обработки новостных аудиопотоков и дальнейшего развития в направлении масштабирования вычислительных узлов и интеграции дополнительных ASR-моделей.

Литература:

1. Speech Recognition in Adverse Conditions: A Review // IEEE Transactions on Audio, Speech, and Language Processing. 2020. С. 1–12.

2. Кузнецов П. А. Методики оценки масштабируемости программных решений // Системный анализ и управление. 2019. С. 32–38.

3. Audio Course: Introduction to Audio Data [Электронный ресурс]. — Режим доступа: https://huggingface.co/learn/audio-course (дата обращения: 16.03.2026).

4. Icecast Streaming Media Server Documentation [Электронный ресурс]. — Режим доступа: https://icecast.org/docs/ (дата обращения: 16.03.2026).

5. Preprocessing Audio Data for Machine Learning [Электронный ресурс]. — Режим доступа: https://huggingface.co/learn/audio-course/chapter1/preprocessing (дата обращения: 16.03.2026).

6. Data Processing Pipeline [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/azure/architecture/data-guide/ (дата обращения: 16.03.2026).

7. Коротких Ю. В. Методы и критерии оценки производительности информационных систем // Вестник Омского университета. 2021. С. 12–18.

8. Литвиненко А. В. Критерии оценки функциональности систем автоматизации предприятий // Информационные технологии в бизнесе. 2020. С. 45–51.

9. Whisper: Robust Speech Recognition via Large-Scale Weak Supervision [Электронный ресурс]. — Режим доступа: https://github.com/openai/whisper (дата обращения: 16.03.2026).

10. T-one: Streaming ASR for Russian Telephony [Электронный ресурс]. — Режим доступа: https://huggingface.co/t-tech/T-one (дата обращения: 16.03.2026).

11. Рулева Е. Д. Комплексный подход при разработке требований к программному продукту // Наука без границ: взгляд молодых ученых. Москва, 2022. С. 146–153.

12. What are Microservices? [Электронный ресурс]. — Режим доступа: https://aws.amazon.com/microservices/ (дата обращения: 18.04.2026).

13. Stream processing with Azure Databricks [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/data/stream-processing-databricks (дата обращения: 18.04.2026).

14. Jurafsky D., Martin J. Speech and Language Processing. 3rd ed. Pearson, 2023. 560 p.

15. Hinton G., Deng L., Yu D. Deep Neural Networks for Acoustic Modeling in Speech Recognition // IEEE Signal Processing Magazine. 2012. С. 82–97.

16. Van Rossum G., Drake F. The Python Language Reference Manual. Python Software Foundation, 2022. 250 p.

17. Why Docker Compose? [Электронный ресурс]. — Режим доступа: https://docs.docker.com/guides/docker-compose/why/ (дата обращения: 18.04.2026).

18. Docker Compose [Электронный ресурс]. — Режим доступа: https://docs.docker.com/guides/docker-compose/ (дата обращения: 18.04.2026).

19. GPU Acceleration for AI Workloads [Электронный ресурс]. — Режим доступа: https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/ (дата обращения: 16.03.2026).

20. Measuring Latency in Distributed Systems [Электронный ресурс]. — Режим доступа: https://cloud.google.com/architecture (дата обращения: 18.04.2026).

21. Kumar A., Hegde P., Vaze R. Breaking the Unit Throughput Barrier in Distributed Systems [Электронный ресурс]. — Режим доступа: https://arxiv.org/abs/2010.07430 (дата обращения: 18.04.2026).

22. Task Success Rate (TSR) Metrics [Электронный ресурс]. — Режим доступа:

https://www.emergentmind.com/topics/task-success-rate-tsr (дата обращения: 18.04.2026).

23. Reliability Testing and Fault Handling in Distributed Systems [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/en-us/azure/architecture/ (дата обращения: 18.04.2026).