ООО «Иксими»

Описание функциональных характеристик программного обеспечения «Открытый квотировано-кластерный оркестратор ИИ (ОКОИИ) на базе архитектуры PBDR "OQOAI:PBDR 5.2"»

г. Зеленодольск

2026 год

Настоящий документ содержит описание функциональных характеристика программного обеспечения «Открытый квотировано-кластерный оркестратор ИИ (ОКОИИ) на базе архитектуры PBDR "OQOAI:PBDR 5.2"»

Назначение и область применения

Программное обеспечение «Открытый квотировано-кластерный оркестратор ИИ (ОКОИИ) на базе архитектуры PBDR "OQOAI:PBDR 5.2"» предназначено и применяется для децентрализованной маршрутизации и управления вычислительными ресурсами на основе политик (Policy-Based Decentralized Routing, PBDR) для задач искусственного интеллекта, обработки запросов и оптимизации распределения вычислительной нагрузки, моделирования, планирования, управления распределенными системами ИИ, на базе реализации интеллектуальной поддержки принятия решений на основе политик и многокритериального расчета значений эффективности узла. ПО используется для построения отказоустойчивого, децентрализованного, эффективного кластера для размещения ИИ-моделей на базе GPU ресурсов

Функциональные задачи

Платформа обеспечивает решение следующих задач:

1. Прием и обработка запросов внешних потребителей

Задача: Обеспечение единой точки входа для всех запросов к LLM-моделям через стандартизированный API.

Описание: ПО принимает входящие запросы на генерацию от внешних приложений и пользователей через OpenAI-совместимый REST API (эндпоинты /v1/chat/completions и /v1/models). Осуществляется валидация входящих параметров (модель, промпт, температура, max_tokens, флаг стриминга), аутентификация и авторизация запросов (API-ключи). Поддерживается потоковая передача ответов (Server-Sent Events / WebSocket) для обеспечения интерактивного взаимодействия с пользователем.

2. Децентрализованная маршрутизация запросов

Задача: Распределение входящих запросов к LLM между вычислительными узлами (серверами или рабочими станциями) без использования единой центральной очереди или единого планировщика.

Описание: Решение о маршрутизации принимается непосредственно на стороне клиента. Каждый клиентский узел независимо опрашивает доступные серверы через Monitor API, собирает их актуальное состояние и вычисляет оптимальный узел для обработки запроса. Такой подход исключает единую точку отказа (SPOF) и обеспечивает линейную масштабируемость системы при росте количества клиентов и вычислительных узлов.

3. Взаимодействие с LLM-серверами

Задача: Обеспечение унифицированного взаимодействия с различными LLM-серверами (Ollama, llama.cpp, vLLM) через их нативные или совместимые с OpenAI API интерфейсы.

Описание: ПО осуществляет обнаружение доступных моделей на каждом вычислительном узле, отслеживание загруженной в VRAM модели, проксирование запросов к LLM-серверу и возврат сгенерированных ответов. Поддерживается как обычный, так и стриминговый режим генерации. Преобразование форматов запросов и ответов выполняется автоматически в зависимости от типа используемого LLM-сервера.

4. Векторная модель расчета стоимости запроса (Cost Vector)

Задача: Многофакторная оценка стоимости обработки запроса на каждом вычислительном узле для принятия оптимального решения о маршрутизации.

Описание: Стоимость обработки запроса на каждом узле представлена вектором из 10 независимых компонентов:

- **c1: Время ожидания** — прогнозируемое время до начала обработки запроса с учетом текущей очереди.
- **c2: Время инференса** — оценочное время генерации ответа на основе производительности узла.
- **c3: Штраф за холодный старт** — штраф за необходимость загрузки модели в VRAM, если она не загружена.
- **c4: Глубина очереди** — нелинейный штраф за длину очереди ожидающих запросов.
- **c5: Загрузка GPU** — квадратичный штраф за высокую загрузку графического процессора.
- **c6: Загрузка CPU** — квадратичный штраф за высокую загрузку центрального процессора.
- **c7: Давление на VRAM** — штраф за недостаток свободной видеопамяти.
- **c8: Тепловой штраф** — штраф за перегрев GPU, снижающий производительность.
- **c9: Бонус за простой** — отрицательная стоимость, поощряющая узлы, давно не использовавшиеся (LRU-принцип).
- **c10: Сетевой штраф** — штраф за сетевую задержку

5. Политики интеллектуальной маршрутизации

Задача: Гибкая настройка приоритетов выбора узла через весовые коэффициенты для каждого компонента вектора стоимости, адаптируемая под различные сценарии использования.

Описание: Политика маршрутизации определяется вектором весов $\Pi \in R^{10}$. Скалярная стоимость для каждого узла вычисляется как скалярное произведение вектора политики и вектора стоимости

Минимальная задержка — приоритет скорости ответа (интерактивные чаты, поддержка).

- **Приоритет загруженной модели** — минимизация переключений моделей (разработка).
- **Энергосбережение** — выбор наиболее холодных узлов (фоновые задачи).
- **Сбалансированная** — равномерное распределение нагрузки.

Администратор может создавать собственные политики, изменяя весовые коэффициенты, и применять их к отдельным устройствам, группам или всему кластеру.

6. Обнаружение и мониторинг доступности узлов

Задача: Автоматическое обнаружение вычислительных узлов в сети и сбор их актуального состояния для принятия решений о маршрутизации.

Описание: Клиентский узел периодически (или по запросу) опрашивает доступные серверы через Monitor API (GET /status), собирая метрики состояния: загрузка GPU/CPU, использование VRAM, температура, длина очереди, загруженная модель, доступные модели. Используется версионный протокол для минимизации сетевого трафика. При обнаружении недоступных узлов они исключаются из процесса маршрутизации.

7. Администрирование клиентских и серверных узлов

Задача: Централизованное управление всеми компонентами кластера через единый веб-интерфейс.

Описание: Администратор через веб-интерфейс PBDR Admin может:

- просматривать список и статус всех узлов кластера (клиентов и серверов);
- проверять доступность узлов и их текущие метрики;
- добавлять и удалять устройства вручную или через автоматическое сканирование сети;
- управлять конфигурациями узлов (чтение, изменение, запись);
- управлять политиками маршрутизации для отдельных устройств, групп или всего кластера.

8. Мониторинг и сбор метрик аппаратных ресурсов узлов

Задача: Сбор в реальном времени детальной информации об аппаратном обеспечении каждого вычислительного узла для принятия решений о маршрутизации и анализа производительности.

Описание: На каждом вычислительном узле (PBDR Server) ПО собирает информацию о:

- GPU (модель, загрузка ядер, использование VRAM, температура, потребляемая мощность, скорость вентилятора) через NVML (NVIDIA) или ROCm (AMD);
- CPU (загрузка в процентах, количество ядер/потоков);

- RAM (общая, используемая, свободная). Собранные метрики публикуются через Monitor API и используются клиентскими узлами для принятия решений о маршрутизации.

9. Управление расширенной конфигурацией серверов и клиентов

Задача: Гибкое управление настройками всех компонентов системы с возможностью группового и массового обновления.

Описание: Администратор может читать и изменять конфигурационные файлы (формат JSON) на отдельных устройствах, группах устройств или на всех устройствах одновременно. Управление включает:

- настройки клиента (список серверов, параметры буферизации, интервалы опроса, политики);
- настройки сервера (порты, максимальная очередь, параметры мониторинга, флаги приема задач);
- параметры мониторинга (интервалы сбора метрик, пороговые значения). Обновление конфигурации выполняется через Admin API с возможностью применения без перезапуска компонентов (горячая перезагрузка).

Функциональные возможности

1. Прием и обработка запросов внешних потребителей генерации запросов к LLM

Обеспечение единой точки входа для всех запросов к LLM-моделям через стандартизированный API.

2. Децентрализованная маршрутизация запросов

Распределение входящих запросов к LLM между вычислительными узлами без использования единой центральной очереди или единого планировщика.

3. Взаимодействие с LLM-серверами

Обеспечение унифицированного взаимодействия с различными LLM-серверами через их нативные или совместимые интерфейсы.

4. Векторная модель расчета стоимости запроса

Многофакторная оценка стоимости обработки запроса на каждом вычислительном узле для принятия оптимального решения о маршрутизации.

5. Политики интеллектуальной маршрутизации

Гибкая настройка приоритетов выбора узла через весовые коэффициенты для каждого компонента вектора стоимости.

6. Обнаружение и мониторинг доступности узлов

Автоматическое обнаружение вычислительных узлов в сети и сбор их актуального состояния для принятия решений о маршрутизации.

7. Администрирование клиентских и серверных узлов

Централизованное управление всеми компонентами кластера через единый веб-интерфейс.

8. Мониторинг и сбор метрик аппаратных ресурсов узлов

Сбор в реальном времени детальной информации об аппаратном обеспечении каждого вычислительного узла.

9. Управление расширенной конфигурацией серверов и клиентов

Гибкое управление настройками всех компонентов системы с возможностью группового и массового обновления.

Входные данные

№	Тип данных	Источник	Формат	Описание
1	**Запрос на генерацию**	Внешнее приложение / пользователь	JSON (OpenAI API)	Содержит модель, список сообщений (промпт), температуру, max_tokens, флаг стриминга
2	**Конфигурация клиента**	Администратор / Файловая система	JSON	Список серверов, порты, политики, параметры буферизации и опроса
3	**Конфигурация сервера**	Администратор / Файловая система	JSON	Порт мониторинга, максимальная очередь, флаги приема задач
4	**Команды администрирования**	Администратор	HTTP/JSON (Admin API)	Запросы на изменение конфигурации, управление политиками, сканирование сети
5	**Метрики оборудования**	Аппаратное обеспечение узла	NVML/ROCm API, psutil	Данные о загрузке GPU/CPU, VRAM, температуре, мощности

Выходные данные

№	Тип данных	Получатель	Формат	Описание
1	**Ответ LLM**	Внешнее приложение / пользователь	JSON (OpenAI API)	Сгенерированный текст, статистика использования токенов
2	**Потоковый ответ LLM**	Внешнее приложение / пользователь	Server-Sent Events / WebSocket	Постепенная передача сгенерированного текста
3	**Состояние узла (статус)**	PBDR Client	JSON (Monitor API)	Метрики узла: загрузка GPU/CPU, VRAM, температура, очередь, загруженная модель
4	**Список узлов**	Администратор	JSON / Web-интерфейс	Перечень всех узлов кластера с их статусом и метриками
5	**Конфигурация узла**	Администратор	JSON	Текущие настройки клиента или сервера
6	**Логи и метрики**	Администратор	Текст / JSON	Журнал событий, трассировка запросов, статистика производительности

Архитектура ПО

Архитектура приложения реализована следующим образом:

1. Пользователь / Приложение формирует запрос на генерацию (инференс) через OpenAI-совместимый API.

2. PBDR Client выполняет децентрализованную маршрутизацию запроса.

3. PBDR Client проксирует запрос на выбранный PBDR Server.

4. PBDR Server получает ответ от LLM и возвращает его клиенту.

5. Администратор управляет кластером через Admin Web UI.

Взаимодействие между компонентами

Этапы взаимодействия

1. Пользователь / Приложение → PBDR Client (OpenAI API)

- Пользователь отправляет HTTP-запрос на эндпоинт /v1/chat/completions
- Запрос содержит параметры: модель, промпт, температура, max_tokens, флаг стриминга
- PBDR Client принимает запрос, валидирует параметры и инициирует процесс маршрутизации

2. PBDR Client → PBDR Server (Monitor API)

- PBDR Client выполняет HTTP-запросы GET /status ко всем известным PBDR Server узлам
- Запрос содержит заголовок If-Version для реализации версионного протокола
- PBDR Server возвращает актуальное состояние узла (метрики) в формате JSON

3. PBDR Client (внутренняя обработка)

- На основе полученных метрик PBDR Client строит **Cost Vector** (вектор стоимости) из 10 компонентов для каждого узла
- Применяет текущую политику маршрутизации (весовой вектор)
- Вычисляет скалярную стоимость для каждого узла
- Выбирает узел с минимальной стоимостью

4. PBDR Client → PBDR Server (проксирование запроса)

- PBDR Client отправляет оригинальный запрос к LLM на выбранный PBDR Server
- PBDR Server принимает запрос, преобразует его в формат, ожидаемый локальным LLM-сервером
- PBDR Server передает запрос на выполнение LLM-серверу

5. PBDR Server → LLM Сервер

- LLM-сервер (Ollama, llama.cpp, vLLM, TGI) обрабатывает запрос
- Генерация ответа происходит в обычном или стриминговом режиме
- Сгенерированный ответ возвращается в PBDR Server

6. PBDR Server → PBDR Client → Пользователь

- PBDR Server возвращает ответ PBDR Client
- PBDR Client передает ответ обратно пользователю в формате OpenAI API
- При стриминговом режиме ответ передается по частям через WebSocket/SSE

7. PBDR Admin (дополнительно)

- Администратор через веб-интерфейс PBDR Admin отправляет команды на PBDR Client и PBDR Server через Admin API
- Команды включают: просмотр статуса узлов, изменение конфигураций, управление политиками, сканирование сети

Требования к техническим средствам (Эксплуатационные характеристики)

Для установки и эксплуатации программного обеспечения «Открытый квотировано-кластерный оркестратор ИИ (ОКОИИ) на базе архитектуры PBDR "OQOAI:PBDR 5.2"» необходимо, чтобы аппаратное обеспечение соответствовало следующим требованиям:

- ЦП: x86_64 архитектура, от 2 ядер
- ОЗУ: от 4 ГБ (зависит от размера используемой языковой модели).
- GPU: Nvidia или AMD
- Дисковая подсистема: SSD от 25 ГБ (зависит от размера загруженной языковой модели).
- ОС: Рекомендуется **Astra Linux, ОСнова.**
  Поддержка:
  - Дистрибутив Linux с поддержкой python 3.8+ (Debian, Ubuntu),
  - Windows 10/11

Описание функциональной части программного обеспечения

Языки программирования программного обеспечения «Открытый квотировано-кластерный оркестратор ИИ (ОКОИИ) на базе архитектуры PBDR "OQOAI:PBDR 5.2"», – JavaScript, Python

Информация, необходимая для установки и эксплуатации

Для установки и правильной эксплуатации «Открытый квотировано-кластерный оркестратор ИИ (ОКОИИ) на базе архитектуры PBDR "OQOAI:PBDR 5.2"» необходимо ознакомиться с информацией, размещенной в Руководстве пользователя.