



Кадровый системный ИТ-интегратор

Инфраструктура для AI-агентов в закрытом контуре SENSE

Локальный инференс · Управляемые агенты · Закрытый контур · Эксплуатация

Агентная платформа — это инженерный программно-аппаратный комплекс, а не набор промптов.

Жельвис В. А. — заместитель начальника отдела внутренней разработки

Нулевой исходящий трафик как архитектурная граница

Требование	Архитектурное следствие
Кадровые данные не покидают контур	Локальный инференс и локальная память
Нулевой выход в интернет	Внутренние реестры, модели, инструменты и обновления
Агент работает с кодом и системами заказчика	Изолированные рабочие пространства и инструменты по ролям
Нужен полный жизненный цикл задачи	Оркестратор, память, аудит, мониторинг
Платформа должна приниматься в эксплуатацию	KPI, тесты, откат, резервное копирование, регламенты

Это не попытка «закрыть облако межсетевым экраном». Это архитектура, изначально построенная вокруг доверенной локальной границы: внутри контура — инференс, оркестрация, память, инструменты, репозитории, песочницы и наблюдаемость.



Сначала — KPI и границы; затем — код и модели

Область	Реализованное требование
Управляемость	≥ 20 одновременно активных агентов
Время создания агента	≤ 2 секунд
Старт оркестрации	≤ 5 секунд
Управляющий API, p95	≤ 500 мс
Ошибки API	$\leq 0,5\%$
Масштабирование сервисов	Без остановки клиентского API
Безопасность	RBAC, аудит, сегментация данных, нулевой исходящий трафик
Эксплуатация	Версионирование, откат, резервное копирование, мониторинг, приёмка

≥ 20

активных агентов
одновременно

≤ 2 с

создание
агента

≤ 5 с

старт
оркестрации

≤ 500 мс

управляющий API
(p95)

$\leq 0,5\%$

ошибки
API

«Активный агент» включает состояния «выполняется», «ожидание», «приостановлен» и «выполнение инструмента»; число одновременных LLM-последовательностей нормируется отдельно профилем vLLM. Управляющий API — API ПАК: создание агентов, запрос состояния, управление жизненным циклом и отмена.

Один узел: инференс, контур управления и агентная среда исполнения

АППАРАТНАЯ ОСНОВА КОНТУРА

NVIDIA DGX Spark / GB10

Процессор	20-ядерный Arm Grace + Blackwell
Память	128 ГБ унифицированной LPDDR5x
Пропускная способность	273 ГБ/с
Вычисления	до 1 PFLOP FP4
Сеть	ConnectX-7 · 200 Гбит/с
Питание	240 Вт БП / 140 Вт TDP GB10

СОСТАВ ПАК НА УЗЛЕ

DGX Spark

- vLLM / Qwen
- шлюз LiteLLM
- агентная среда исполнения
- PostgreSQL + Prometheus
- MCP-шлюз
- рабочие пространства Coder

Один корпус на первом этапе — преимущество: проще физическая защита, резервирование конфигурации, наблюдаемость, контроль доступа и приёмка. Единая память задаёт бюджет памяти, ConnectX-7 — аппаратный задел для роста.

Qwen3.6-35B-A3B NVFP4: проектируем не только ток/с

МОДЕЛЬ

Qwen3.6-35B-A3B NVFP4

35B параметров всего / 3B активных
МоЕ: 256 экспертов, 8 выбираемых + 1 общий
нативный контекст 262K
YaRN-профиль до ~1.01M — не по умолчанию

РАСЧЁТ ПАМЯТИ

Бюджет единой памяти
веса модели + KV-кэш + префилл / исполнение
+ сервисы CPU / ОС + запас на пиковые нагрузки
= **профиль одновременности и контекста**

ДЛИНА КОНТЕКСТА → ПРЕФИЛЛ И ЗАДЕРЖКА · НАШИ ЗАМЕРЫ (DGX Spark, c = 1)

6 180

ток/с при 4K
0,7 с до первого ответа

5 865

ток/с при 16K
2,8 с до первого ответа

4 203

ток/с при 64K
15,6 с до первого ответа

3 424

ток/с при 100K
29,2 с до первого ответа

Длинный контекст — не «чуть дороже», а отдельный бюджет: рост глубины увеличивает KV-кэш и префилл и снижает допустимую конкуренцию. Проектируем общий бюджет памяти, а не одну цифру ток/с.

Четыре контура ответственности

1. Пользовательский слой

IDE · OpenCode · Web · curl · REST API



2. Контур управления

LiteLLM · Ключи API · RBAC · Квоты · Аудит · Prometheus



3А. Контур инференса

vLLM · Qwen · KV-кэш
Кэш префиксов · планировщик



3В. Контур агентов

Менеджер · Оркестратор
Память · взаимодействие



4. Контур инструментов и исполнения

MCP-шлюз · GitLab · Tracker · Figma · рабочие пространства Coder

Контур состояния: PostgreSQL · хранилище памяти · события аудита · резервные копии — компоненты без состояния реплицируются, у компонентов с состоянием есть явные точки хранения, резервирования и восстановления.

Агент = конечный автомат + квота + политика + аудит



ЛИМИТЫ АГЕНТА

- бюджет токенов
- число итераций
- квота памяти
- предел времени выполнения
- вызовы MCP
- квота пространства

ОРКЕСТРАТОР

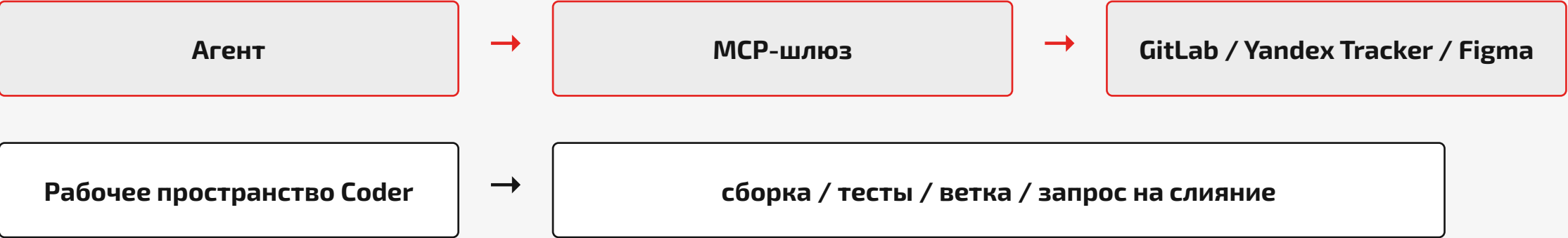
DAG задач → назначение ролей → повтор / переназначение / откат

Основная доменная логика, которую нельзя получить простой установкой сервера инференса.

Невозможно безопасно эксплуатировать компонент, у которого нет измеримых границ: у каждого агента фиксируются лимиты, права и полный журнал действий.

МСП и песочницы: агент получает только нужные права

Роль	Права	Среда
Аналитик	Tracker, память только на чтение	Чтение задач и документации
Разработчик	GitLab, рабочее пространство Coder, инструменты для кода	Сборка, тесты, ветки, запрос на слияние
Дизайнер	Figma, Tracker	Макеты и задачи
Рецензент	Чтение GitLab, отчёты о тестах, контекст задачи	Код и результаты тестов



Права выдаются на уровне роли и сценария, а не «по факту наличия интеграции». Исполнение кода — в изолированном рабочем пространстве Coder: воспроизводимая среда, заданные квоты, без секретов промышленной среды; сеть и Docker — только там, где нужны.

От задачи до результата: восемь контролируемых шагов



На каждом шаге: идентификация · бюджет · тайм-аут · аудит · отмена

Не только штатный сценарий:
отмена, тайм-аут, управляемая ошибка

Пользователь получает артефакты, а не только текст: план, изменения, результаты проверок, ссылки на задачи или запрос на слияние.

ПАК готов к эксплуатации: наблюдать, восстановить, принять

НАБЛЮДАЕМОСТЬ

- ▶ Prometheus: запросы, токены, TTFT, задержки, ошибки
- ▶ Логи сервисов и события аудита
- ▶ Метрики очередей, процессов, рабочих пространств
- ▶ Алерты и проверки работоспособности

БЕЗОПАСНОСТЬ

- ▶ Нулевой исходящий трафик, RBAC, области ключей, MCP по ролям
- ▶ Сегментация: пользователь / агент / администратор
- ▶ Секреты вне Git и образов, ротация
- ▶ Политики хранения и маскирования

ЭКСПЛУАТАЦИЯ

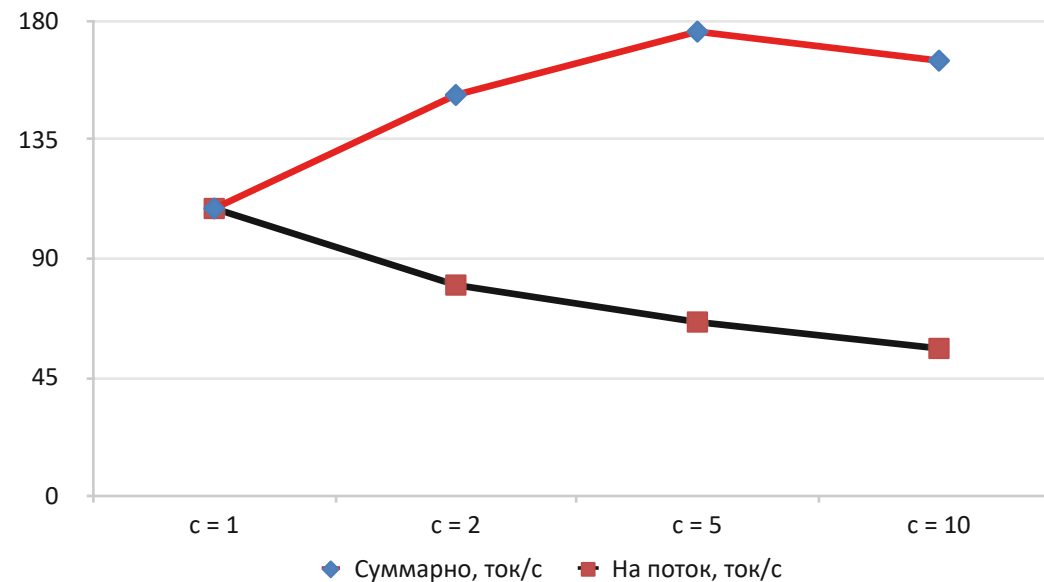
- ▶ Версионированные манифесты и контейнерные образы
- ▶ Откат на валидированный артефакт развёртывания
- ▶ Резервное копирование PostgreSQL + проверка восстановления
- ▶ Функциональные, нагрузочные, на безопасность и интеграционные испытания

Приёмка: сценарии → метрики → протокол → решение комиссии

Производительность агента — это задержка всего процесса

Метрика KPI ПАК	Требование
Активные агенты	≥ 20
Параллельные задачи средней сложности	8–12
Создание агента	≤ 2 сек
Старт оркестрации	≤ 5 сек
Управляющий API, p95	≤ 500 мс
Неуспешные запросы	$\leq 0,5\%$
Масштабирование API-сервисов	Без остановки клиентского API

Конкурентность → суммарная и по-поточная скорость



TTFR при префилле 2048: 373 мс → 819 мс → 1 718 мс → 3 710 мс

DGX Spark / GB10 · nvidia/Qwen3.6-35B-A3B-NVFP4 · профили pp2048 / tg128 · средние значения по замерам. Полная сетка, отклонения и методика — резервный слайд 4. Определение «средней сложности» задачи зафиксировать в методике испытаний до репетиции.

Масштабирование — последовательность инженерных шагов

Этап	Архитектурный ход	Что масштабируется
Сейчас	1 × DGX Spark, Docker Compose	Полный локальный контур
Этап 1	Репликация сервисов vLLM	Больше независимых потоков инференса
Этап 2	Выделение нагрузок агентов, исполнения и инструментов	Изоляция нагрузки и отказов
Этап 3	Kubernetes + декларативное развёртывание	Управление размещением и жизненным циклом
Этап 4	Несколько DGX Spark через ConnectX-7	Отдельно валидируемый распределённый инференс

Terraform / IaC: описываем, тестируем и воспроизводим окружение

Kubernetes не заменяет проектирование распределённого инференса: отдельно определяются топология, реплики, сервисы с состоянием, коллективная коммуникация и домены отказа. Аппаратный задел — ConnectX-7; главный актив интегратора — инфраструктура как код.

Инженерия агентных систем начинается до первой генерации

**Закрытый контур → локальный инференс → управляемые агенты →
изолированные инструменты → наблюдаемость → эксплуатация → рост**

01

Нулевой исходящий трафик определяет архитектуру, а не только сетевую политику.

02

Агент должен иметь конечный автомат, квоты, политику и аудит.

03

В инференсе важны KV-кэш, префилл, одновременность и хвостовые задержки — не только ток/с.

04

ТЗ, испытания и эксплуатация должны появляться одновременно с AI-функциональностью.

SENSE: агентная платформа как промышленный ПАК

Профиль обслуживания vLLM: параметры фиксируются с версией образа

РАБОЧИЙ ПРОФИЛЬ

Модель : Qwen3.6-35B-A3B NVFP4
Среда : vLLM, ARM64 / GB10
Внимание : FlashInfer
Бэкенд MoE : Marlin / под железо
KV-кэш : FP8
MTP : драфт на 3 токена
Почастичный префилл : включён
Кэширование префикса : включено, где можно
gri-memory-util. : 0.5 (рабочий профиль)

ВАЖНО

Для DGX Spark наш рабочий профиль ограничивает gri-memory-utilization значением 0.5 вследствие общей памяти CPU/GPU.

Все альтернативные профили требуют отдельной валидации на пиковое потребление памяти, OOM и задержки.

Параметры профиля меняются только вместе с версией образа и проходят регрессионную проверку.

Слайд 5 не выводит gri-memory-utilization: рабочее значение 0.5 обсуждается только как экспериментальный профиль с подтверждённой методикой.

Почему 1M контекста не является режимом по умолчанию

Режим	Когда применяется	Компромисс
Нативный контекст 262K	Основной агентный режим	Предсказуемый бюджет памяти и задержки
~524K YaRN	Крупные спецификации и репозитории	Больше KV-кэш и префилла
~1.01M YaRN	Редкие сверхдлинные задачи	Сильное влияние на одновременность и сквозные задержки

YaRN — это не «бесплатное увеличение окна». Статический YaRN изменяет масштабирование независимо от длины входа, поэтому его не включают без реальной потребности в длинных промптах. Наши замеры: с ростом глубины префилл падает с 6 180 до 3 424 ток/с, а задержка до первого ответа вырастает с 0,7 до 29 с (резервный слайд 4). Для нагрузок ~524K множитель масштабирования уменьшается относительно профиля на ~1M.

262K

нативный — основной режим

~524K

YaRN — крупные документы

~1.01M

YaRN — редкие задачи

Что означает 20 активных агентов

Состояние	LLM GPU	Инструменты	Активный агент
Выполняется: инференс	Да	Нет	Да
Ожидание: MCP / API	Обычно нет	Да	Да
Работа в пространстве	Обычно нет	Да	Да
Чтение памяти	Обычно нет	Нет / частично	Да
Приостановлен	Нет	Нет	Да
Завершён / Ошибка	Нет	Нет	Нет

Активные агенты ≠ одновременные LLM-последовательности

Число одновременных LLM-последовательностей — отдельный параметр профиля обслуживания (max_num_seqs, конкурентность батча) и нормируется не KPI «20 агентов», а бюджетом памяти: веса, KV-кэш и глубина контекста. Показатели разведены намеренно, чтобы не смешивать управляемость платформы с пропускной способностью инференса.

Методика и результаты наших измерений

Оборудование: NVIDIA DGX Spark / GB10, единая LPDDR5x · Модель: nvidia/Qwen3.6-35B-A3B-NVFP4 · Стек: vLLM в контуре SENSE · Профили: pp2048 (префилл 2048) / tg128 (генерация 128) / ctx_* (глубина контекста d) / c — одновременные запросы · Отчёт: среднее ± отклонение

Профиль	Результат
pp2048, c = 1	5 529 ± 30 префилл-ток/с; 373 мс до первого ответа
tg128, c = 1	109,3 ± 4,2 ток/с
tg128, c = 2	151,6 ± 5,0 суммарно / 79,6 ± 4,1 на поток
tg128, c = 5	175,5 ± 8,6 суммарно / 66,4 ± 14,9 на поток
tg128, c = 10	165,0 ± 3,5 суммарно / 55,7 ± 14,8 на поток; 3 710 ± 2 210 мс TTFR
ctx_pp @ 4K, c = 1	6 180 ± 14 ток/с; 0,7 с до первого ответа
ctx_pp @ 16K, c = 1	5 865 ± 238 ток/с; 2,8 с
ctx_pp @ 64K, c = 1	4 203 ± 7 ток/с; 15,6 с
ctx_pp @ 100K, c = 1	3 424 ± 8 ток/с; 29,2 с
ctx_tg @ 100K, c = 1	100,3 ± 7,8 ток/с
ctx_tg @ 100K, c = 10	8,8 ток/с на поток, сильный разброс

КАК ЧИТАТЬ

1. Суммарная скорость перестаёт расти после c = 5 — узел упирается в память и батч.
2. Скорость потока падает с 109 до 56 ток/с: агенты ждут дольше, растёт хвостовая задержка.
3. Длинный контекст бьёт по задержке сильнее, чем по скорости: префилл на 100K ≈ 30 с.
4. Разброс TTFR при c = 10 велик (3 710 ± 2 210 мс) — называть порядок величины.

Программа испытаний

Группа испытаний	Проверяемые объекты	Критерий
Функциональные	Менеджер агентов, Оркестратор, память, интерфейс	Полный жизненный цикл, ошибки, отмена, возобновление
Интеграционные	LiteLLM, vLLM, MCP, Coder, GitLab, Tracker	Сквозной сценарий разработки
Нагрузочные	API, оркестрация, очередь инференса, вызовы инструментов	KPI, p95/p99, отсутствие деградации, управляемый отказ
Безопасность	RBAC, исходящий трафик, секреты, изоляция рабочих пространств	Отсутствие неразрешённых путей и превышения прав
Эксплуатационные	Резервное копирование, восстановление, откат, мониторинг	Восстановление и наблюдаемость по регламенту

Неразрешённых путей нет по результатам испытаний — и это проверяемо протоколом. Платформа принимается не тогда, когда модель отвечает «хорошо», а когда есть воспроизводимый способ доказать, что после изменения система осталась приемлемой.

Почему LiteLLM поверх vLLM

Компонент	Основная ответственность
vLLM	Обслуживание модели, батчинг, KV-кэш, планирование, инференс, совместимый с OpenAI
LiteLLM	Единая внешняя модельная витрина, виртуальные ключи, ограничения частоты, квоты, учёт, маршрутизация
Среда исполнения агентов	Процессы, агенты, политики, конечный автомат, память, выдача прав инструментам
MCP-шлюз	Контракт и политика доступа к инструментам

vLLM обслуживает модель.

LiteLLM управляет доступом к модельному сервису.

Среда исполнения агентов управляет их поведением.

Вопрос о Kubernetes

Kubernetes — это следующий уровень эксплуатации сервисов, а не кнопка «сделать распределённый инференс».

На первом этапе мы масштабируемся через реплики инференса и выделение нагрузок.

РАСПРЕДЕЛЁННЫЙ ИНФЕРЕНС ТРЕБУЕТ ОТДЕЛЬНОГО ПРОЕКТИРОВАНИЯ:

стратегия параллелизма

коллективная
коммуникация

топология

сетевая подложка

размещение состояния

домены отказа

валидация производительности

Атакующая поверхность — комбинация «права + инструменты + память»

Вектор	Мера
Инъекция в документ, страницу или тикет, которые читает агент	Данные не являются командами; проверка по политике на стороне инструмента, а не доверие к выводу модели
Избыточные права	Выдача прав по роли, по сценарию, а не по факту наличия интеграции
Экспфильтрация через инструменты	Нулевой исходящий трафик, список разрешённых операций, квоты, полный аудит
Самовольное или разрушающее изменение кода	Изолированное рабочее пространство, ветки, проверка, откат, отсутствие секретов промышленной среды
Цикл, деградация, разгон счётчика	Итерации, предельное время, бюджет токенов, лимит вызовов MCP
Социальная инженерия через агента в пользу пользователя	Отказ от действий вне множества разрешённых действий роли; пользователь получает предложение, а не выполнение

Никакая изоляция не объявляется «абсолютной». Мы говорим: неразрешённых путей нет по результатам испытаний (резервный слайд 5), и это проверяемо протоколом. Не подтверждённые формулировки вида «изоляция абсолютна» в доклад не выносятся.

Нулевой исходящий трафик и обработка персональных данных

Вопрос из зала	Как закрыто
Куда уходят персональные данные сотрудников?	Нигде: локальный инференс, локальная память, нулевой исходящий трафик (слайды 2 и 10)
Кто что может сделать?	RBAC, области действия ключей, сегментация пользователь / агент / администратор, MCP по ролям
Как доказать, что агент сделал именно X?	События аудита со сроками хранения и маскированием, версии манифестов, протокол испытаний
Как оценить защищённость?	Программа испытаний на безопасность, включая попытки превышения прав (резервный слайд 8)
Что показать при приёмке?	Сценарии → метрики → протокол → решение комиссии (слайд 10)

Перечень применимых нормативных документов (152-ФЗ, требования ФСТЭК России к защите персональных данных, методики оценки — ГОСТ Р 57580) уточняется под контур конкретного заказчика. На конференции называть общий перечень, конкретные реквизиты актов и договоров не приводить.

Обновление модели — это релиз, а не «новые веса»

Область	Как управляется
Качество ответов	Регрессионный набор сценариев (эталонный набор); приёмка по метрикам, а не по впечатлению
Обновление модели или промптов	Новая версия → повторное прохождение испытаний (резервный слайд 5) → развёртывание или откат
Регрессия инструментов	Проверка контура вызова инструментов после каждого обновления среды исполнения и парсеров
Стоимость узла	Энергопотечение и тепловой пакет узла (слайд 4), ресурс под рабочее пространство, труд на обновление и ротацию ключей — считаются на этапе приёмки
Что не обещаем	Не заявляем «кластер готов», «изоляция абсолютна», «интеграция бесплатна»

Платформа принимается не тогда, когда модель отвечает «хорошо», а когда есть воспроизводимый способ доказать, что после изменения система осталась приемлемой.