

МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, КОМПЛЕКСОВ И КОМПЬЮТЕРНЫХ СЕТЕЙ/MATHEMATICAL SOFTWARE FOR COMPUTERS, COMPLEXES AND COMPUTER NETWORKS

DOI: https://doi.org/10.60797/itech.2026.11.7 EDN: CPUTFP

МЕХАНИЗМЫ ОТКАЗОУСТОЙЧИВОСТИ И ВОССТАНОВЛЕНИЯ В РАСПРЕДЕЛЁННЫХ СИСТЕМАХ AI-ИНФЕРЕНСА

Научная статья

Кротов А.Д.[1, *]
[1] ORCID : 0009-0002-0795-6200;
[1] Яндекс, Москва, Российская Федерация

* Корреспондирующий автор (san.crot[at]yandex.ru)

Предложена: 04.05.2026; Принята: 06.07.2026; Опубликована: 14.07.2026

Аннотация

Статья посвящена систематизации механизмов отказоустойчивости и восстановления в распределённых системах AI-инференса с учётом современных облачных, бессерверных и периферийных архитектур. Актуальность исследования связана с ростом нагрузки на сервисы инференса больших моделей при ограниченных инженерных ресурсах и жёстких требованиях к доступности и задержкам отклика. Новизна работы выражена в сопоставлении подходов к управлению состоянием и деградацией качества в системах ServerlessLLM, DéjàVu, FailLite, Torpor, Hoplite и в обобщении решений по мониторингу, диагностике и обработке отказов в ML-сервисах. В рамках исследования описаны архитектурные шаблоны репликации, потоковой обработки состояний и частичного восстановления, изучены стратегии проектирования бессерверных платформ и облачно-нативных стеков для инференса. Особое внимание уделено балансу между ресурсной эффективностью и глубиной механизмов отказоустойчивости. Работа ставит цель выработать рекомендации по построению надёжной инфраструктуры инференса для распределённых AI-систем. Для её достижения задействованы сравнительный анализ, логическая реконструкция архитектур, синтез концептуальной модели и обобщение результатов известных исследований. В заключении изложены выводы о применимости рассмотренных решений для промышленных серверных систем. Статья предназначена для инженеров и исследователей, проектирующих инфраструктуру инференса в облачных, гибридных и периферийных средах.

Ключевые слова: распределённые системы, AI-инференс, отказоустойчивость, восстановление, инференс в бессерверной среде, облачно-нативная архитектура, LLM-сервинг, репликация, деградация качества, распределённое состояние.

FAULT TOLERANCE AND RECOVERY MECHANISMS IN DISTRIBUTED AI INFERENCE SYSTEMS

Research article

Krotov A.D.[1, *]
[1] ORCID : 0009-0002-0795-6200;
[1] Yandex, Moscow, Russian Federation

* Corresponding author (san.crot[at]yandex.ru)

Suggested: 04.05.2026; Accepted: 06.07.2026; Published: 14.07.2026

Abstract

This article systematizes fault tolerance and recovery mechanisms in distributed AI inference systems, taking into account modern cloud, serverless, and edge architectures. The relevance of this research is due to the increasing load on large-scale model inference services, limited engineering resources, and stringent availability and response time requirements. The novelty of this work lies in its comparison of approaches to state and quality degradation management in ServerlessLLM, DéjàVu, FailLite, Torpor, and Hoplite systems, as well as in its generalization of solutions for monitoring, diagnostics, and failure handling in ML services. This study describes architectural patterns for replication, stream processing, and partial recovery, and examines design strategies for serverless platforms and cloud-native inference stacks. Particular attention is paid to the balance between resource efficiency and the depth of fault-tolerance mechanisms. The goal of this study is to develop recommendations for building a reliable inference infrastructure for distributed AI systems. This is achieved through a comparative analysis, logical reconstruction of architectures, synthesis of a conceptual model, and a synthesis of existing research findings. The conclusions include findings on the applicability of the considered solutions to industrial backend systems. This article is intended for engineers and researchers designing inference infrastructure in cloud, hybrid, and edge environments.

Keywords: distributed systems, AI inference, fault-tolerance, recovery, serverless inference, cloud-native architecture, LLM serving, replication, quality degradation, distributed state.

Введение

Рост доли AI-сервисов в составе корпоративных и пользовательских приложений привёл к тому, что подсистемы инференса стали критическим уровнем инфраструктуры. Нагрузочные профили таких подсистем характеризуются

всплесками запросов, неравномерным распределением трафика по моделям, высокой стоимостью вычислений на GPU и жёсткими ограничениями по задержке ответа. При этом команды сопровождающих инженеров часто небольшие, что усиливает требования к автоматизации отказоустойчивости и восстановлению сервисов без ручного вмешательства администратора.

Цель исследования состоит в формировании целостной картины механизмов отказоустойчивости и восстановления, применяемых в распределённых системах AI-инференса, с ориентацией на практическое проектирование серверной инфраструктуры. Для достижения цели решаются три задачи. Первая задача — классифицировать механизмы отказоустойчивости и восстановления, применяемые в современных системах инференса, по уровням: архитектура сервиса, управление состоянием моделей, эксплуатационный мониторинг и диагностика. Вторая задача — проанализировать конкретные решения в типичных платформах (облачно-нативная архитектура предиктивного моделирования, инференс в бессерверной среде LLM, отказоустойчивый сервинг на периферии, GPU-ориентированные, универсальные механизмы распределённой отказоустойчивости) и выделить устойчивые инженерные приёмы. Третья задача — предложить концептуальную референсную схему инфраструктуры AI-инференса, ориентированную на малочисленные команды, обслуживающие множество моделей, с разграничением обязанностей между уровнем оркестрации, уровнем сервинга моделей и уровнем наблюдаемости.

Новизна работы заключается в том, что механизмы отказоустойчивости, описанные в исследованиях по инференсу в бессерверной среде, LLM-сервингу, периферийным системам и облачно-нативным архитектурам, рассмотрены под единым углом зрения — устойчивое функционирование подсистем инференса при отказах данных, моделей, вычислительной инфраструктуры и элементов оркестрации. Дополнительно проведена интерпретация этих решений с точки зрения инженерной практики разработки серверной части приложений: выделены требования к структуре API, к организации хранения состояний и к стратегиям деградации качества, пригодным для внедрения в промышленных системах, обслуживающих сотни AI-сервисов силами ограниченной команды.

Материалы и методы

В качестве материалов использован ряд научных и технических публикаций, охватывающих разные классы распределённых систем инференса. M.T.T. Bajwa с соавторами описывает облачно-нативную архитектуру крупномасштабного предиктивного моделирования на основе микросервисов, контейнеризации и оркестрации, где обеспечиваются масштабирование и высокая доступность моделей за счёт динамического распределения нагрузки и автоматического управления жизненным циклом сервисов [1]. Y. Fu с коллегами анализирует инференс в бессерверной среде больших языковых моделей и предлагает систему ServerlessLLM с многоуровневым хранением контрольных точек вблизи GPU-узлов, ускоряющим загрузку моделей при колебаниях нагрузки и отказах отдельных узлов [2]. A. Gogineni рассматривает применение моделей глубокого обучения и классических алгоритмов машинного обучения для повышения отказоустойчивости распределённых систем за счёт обнаружения аномалий по телеметрическим данным. В работе упоминаются архитектура VGG16, алгоритмы на основе деревьев решений, метод наивного Байеса и алгоритм случайного леса, используемые для классификации отказов и отклонений в поведении инфраструктуры [3]. L.S. Myllyaho с соавторами проводит исследование архитектурных решений, повышающих устойчивость ML-систем к сбоям, и описывает паттерны ввода-вывода, мониторинга и ограничений бизнес-логики, снижающие влияние неверных или деградировавших моделей [4]. F.F.B. Santos исследует надёжность облачных ML-сервисов методом инъекции ошибок в данные, оценивая реакцию сервисов на различные виды искажения входов и влияние таких искажений на качество и справедливость предсказаний [5]. F. Strati с коллегами предлагает систему DéjàVu для масштабного сервинга LLM, использующую потоковую обработку кэша структуры «ключ–значение» (KV-кэша), раздельную обработку входного текстового запроса пользователя (промпт) и токенов и репликацию состояния для быстрой реакции на отказы и эффективного использования GPU-памяти [6]. L. Wang с соавторами обобщает исследования в области инференса в бессерверной среде, уделяя внимание управлению ресурсами, масштабированию, обеспечению задержки и надёжности в среде функций как сервиса [7]. L. Wu с коллегами описывает систему FailLite для отказоустойчивого сервинга моделей на периферии с гетерогенной репликацией, разделением на тёплые и холодные реплики и прогрессивным восстановлением, ориентированным на минимизацию времени восстановления при жёстких ресурсных ограничениях [8]. M. Yu с соавторами представляет бессерверную платформу Torpor с GPU-ориентированным планированием, поздним связыванием и перестановкой моделей между GPU, что снижает стоимость и повышает устойчивость к перегрузкам и временной недоступности ресурсов [9]. S. Zhuang в диссертационной работе предлагает архитектуру Hoplite и другие механизмы эффективной отказоустойчивости в распределённых системах, включая устойчивую коллективную коммуникацию и разделение путей выполнения и восстановления в системах рабочих процессов [10].

Для достижения цели исследования применён сравнительный анализ архитектур и механизмов отказоустойчивости, структурирование механизмов по уровням системы, логический и системный подход к выведению общих принципов проектирования, а также аналитический обзор результатов, представленных в указанных источниках. Использовались приёмы контент-анализа описаний архитектур, классификация видов отказов и связанных с ними механизмов обнаружения и восстановления, а также синтез концептуальной модели инфраструктуры AI-инференса, объединяющей сильные стороны рассмотренных решений.

В статье сохраняются англоязычные обозначения KV-кэш, service mesh, health check, readiness probe, late binding, SLO и TTFT, поскольку они закреплены в технической литературе по распределённому сервингу моделей.

Результаты

Анализ работ, посвящённых распределённым системам инференса, показывает, что устойчивость таких систем формируется через сочетание архитектурных решений по управлению состоянием, репликацией и деградацией качества. Облачно-нативные архитектуры предиктивного моделирования, описанные M.T.T. Bajwa и коллегами,

опираются на микросервисную декомпозицию, контейнеризацию и оркестраторы, такие как Kubernetes, что позволяет гибко распределять сервисы инференса по узлам кластера и компенсировать отказы через перезапуск подов, автоматическое перераспределение трафика и механизмы проверки работоспособности [1]. В таких архитектурах отказоустойчивость усиливается согласованной конфигурацией сервисной сети, компонентов распределения входящего трафика и систем наблюдаемости, что имеет значение для крупных инсталляций, где одновременно обслуживается множество моделей с различными профилями нагрузки.

ServerlessLLM, предложенная Y. Fu с соавторами, демонстрирует подход к работе с состоянием больших моделей инференса в бессерверной среде [2]. Система использует многоуровневую иерархию хранения контрольных точек и связанных артефактов: от локального при-GPU хранилища до удалённых репозиториев, оптимизированных под потоковую загрузку. Загрузка модели организована таким образом, чтобы уменьшить зависимость от удалённых хранилищ и сократить время холодного старта при миграции функций между узлами или отказе отдельных серверов. Механизмы повторной инициализации моделей сочетаются со стратегиями балансировки нагрузки, что обеспечивает устойчивое поведение при всплесках запросов, характерных для генеративных LLM-сервисов.

Система DéjàVu, разработанная F. Strati и коллегами, сосредоточена на сервинге генеративных LLM в распределённой GPU-среде и решает три задачи: сокращение простоев стадий конвейерной обработки при чередовании обработки промпта и генерации токенов, снижение избыточного использования памяти GPU и ускорение восстановления после отказов [6]. Для этого вводится библиотека потоковой обработки KV-кэша, позволяющая отделить обработку промпта от генерации, распределить эти операции между группами GPU и передавать кэш между узлами без полной перезагрузки модели. Репликация состояния токенизации и KV-кэша при отказе узла поддерживает возобновление генерации на другом узле, сокращает потерю уже выполненных вычислений и сохраняет приемлемую задержку пользовательского запроса. Подход DéjàVu показывает, что точная работа с промежуточным состоянием модели снижает зависимость системы от полной репликации и тяжёлого чекпоинтинга.

Показательно, что линия работы с промежуточным состоянием генеративных моделей получила развитие и в open-source среде. В публикации LMSYS за сентябрь 2025 года описан SGLang HiCache — иерархический механизм KV-кэширования, где состояние сессии распределяется между GPU-памятью, памятью хоста и внешними слоями хранения, а контроллер управляет подгрузкой, резервным сохранением, prefetch и политиками записи между уровнями памяти [11]. Авторы представляют HiCache прежде всего как средство снижения TTFT и повышения пропускной способности, но в инженерном чтении такая схема уменьшает зависимость сервинга от единственного горячего слоя KV-состояния и снижает цену повторных вычислений при вытеснении кэша, перепланировании запроса или локальной недоступности части памяти. Open-source практика, следовательно, подтверждает, что подходы, близкие к DéjàVu, вышли за пределы исследовательской постановки и перешли в плоскость прикладной эксплуатации.

L. Wu с соавторами в системе FailLite предлагают иной путь: управляемую деградацию качества и гетерогенную репликацию для периферийных узлов [8]. Вместо полного дублирования тяжёлых моделей, недостижимого в условиях ограничений по памяти и вычислительным ресурсам, FailLite вводит набор облегчённых моделей-заменителей, обученных воспроизводить поведение эталонной модели с допустимой потерей точности. Часть таких моделей поддерживается в тёплом состоянии, что позволяет быстро переключаться на них при отказе основной модели или перегрузке узла, тогда как холодные реплики загружаются по мере необходимости. Прогрессивное восстановление организует последовательный переход от простейших моделей-заменителей к более точным, по мере освобождения ресурсов. Эксперименты демонстрируют среднее время восстановления порядка сотен миллисекунд при падении точности менее 1%, что соответствует требованиям многих приложений, чувствительных к задержкам отклика [8]. Такой подход формирует полезный шаблон для систем, где критична непрерывность сервиса при ограниченных ресурсах.

На рисунке 1 приведена обобщённая архитектура системы FailLite, иллюстрирующая разделение на слой диспетчеризации запросов, пул основных моделей, пул облегчённых реплик и компонент, отвечающий за принятие решений о прогрессивном восстановлении [8].

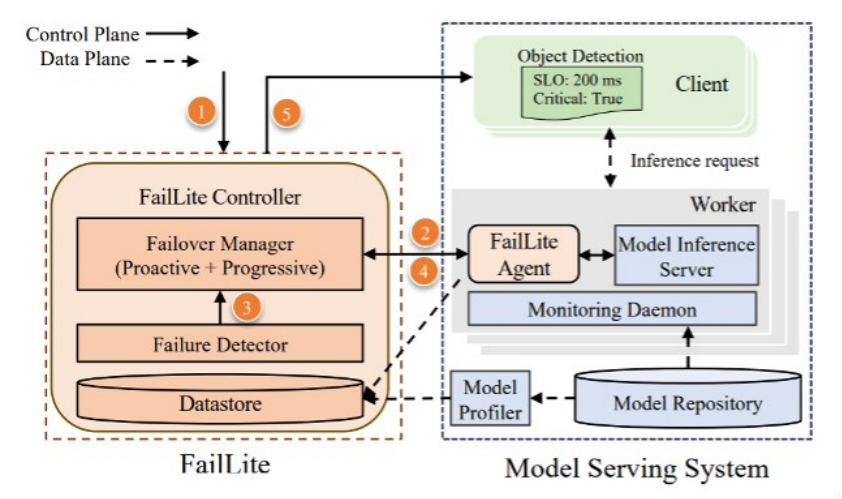

Рисунок 1 - Обобщённая архитектура отказоустойчивого сервинга моделей FailLite
DOI: https://doi.org/10.60797/itech.2026.11.7.1

Примечание: источник [8]

Бессерверная платформа Torpor, предложенная M. Yu с соавторами, дополняет картину на уровне управления ресурсами GPU [9]. Torpor ориентирован на сценарий, где в рамках одной платформы размещается множество функций инференса, каждая со своими требованиями к задержке и нагрузке. Платформа хранит модели в оперативной памяти и выполняет их динамическую загрузку на GPU при поступлении запросов (late binding с перестановкой моделей). Для снижения задержек используются асинхронная переадресация API-вызовов, совместное использование рантайма GPU, конвейерное выполнение и алгоритмы планирования, учитывающие интерференцию нагрузок [9]. Такой подход повышает устойчивость к перегрузкам и временной недоступности отдельных GPU, поскольку планировщик обладает возможностью гибко перераспределять функции по устройствам, сохраняя заданные SLO по задержке.

Работа S. Zhuang демонстрирует, как универсальные механизмы отказоустойчивой коммуникации и управления задачами помогают удерживать устойчивость широкого класса приложений, включая сервинг ансамблей моделей [10]. Hoplite, описанный в этой диссертации, реализует распределённое объектное хранилище с гибкой коллективной коммуникацией, где передача данных и задачи перераспределяются с учётом текущих отказов и перегрузок [10]. За счёт тонкой фрагментации данных и конвейеризации коллективных операций достигается устойчивое поведение при выпадении отдельных узлов, а уровень системных библиотек скрывает детали реализации от прикладного кода. В той же работе предлагается ExoFlow, отделяющий единицы выполнения и восстановления, что позволяет поддерживать семантику ровно-одного выполнения рабочих процессов при сниженных накладных расходах на чекпоинты [10]. Это создаёт основу для построения надёжных конвейеров инференса, где шаги обработки запросов и обновления моделей связываются в рабочих процессах с контролируемой семантикой восстановления.

Исследование L.S. Myllyaho и коллег дополняет архитектурный уровень анализом поведения ML-систем при сбоях и описанием шаблонов проектирования, уменьшающих риск некорректной работы при деградации моделей [4]. Авторы фиксируют распространённые приёмы: ограничения на диапазоны допустимых входов, фильтрация редких или аномальных значений, бизнес-правила для выходов моделей, дублирование моделей с разной сложностью и мониторинг распределения входных данных [4]. Эти приёмы применимы к подсистемам инференса как дополнительные линии защиты поверх инфраструктурных механизмов репликации и восстановления.

Работа F.F.B. Santos дополняет картину на уровне данных, показывая влияние дефектов входной информации на надёжность облачных ML-сервисов [5]. Путём систематической инъекции искажений изображений исследуется устойчивость сервисов компьютерного зрения и влияние таких искажений на точность и справедливость решений [5]. Результаты свидетельствуют, что даже при инфраструктурной устойчивости к отказам узлов и сетевых компонентов система восприимчива к «тихим» ошибкам, связанным с деградацией данных. Это подчёркивает необходимость совместного применения механизмов контроля качества данных и архитектурных решений по отказоустойчивости для подсистем инференса.

A. Gogineni описывает подход к повышению устойчивости распределённых систем на основе глубокого обучения; в работе рассматриваются архитектура VGG16 и классические алгоритмы машинного обучения, включая деревья решений, «наивный» байесовский классификатор и алгоритм случайного леса, применяемые для обнаружения и классификации отказов [3]. Сравнение показывает, что глубокие модели достигают высокой точности и F1-метрик при распознавании отказов, что позволяет использовать их в качестве интеллектуального компонента диагностики инфраструктуры, выявляющего отклонения до перехода сбоя в состояние недоступности сервиса [3]. Такой подход органично дополняет архитектурные решения ServerlessLLM, DéjàVu, FailLite и Hoplite, где требуется своевременная информация о состоянии узлов и сервисов.

Наконец, работа L. Wang с коллегами обобщает исследования по инференсу в бессерверной среде и подчёркивает значение ресурсов, масштабирования, задержек и надёжности для таких систем [7]. В обзоре названы направления исследований: оптимизация запуска функций, интеграция GPU и специализированных ускорителей, уменьшение

накладных расходов на коммуникацию и обеспечение гарантированной доступности функций в условиях переменной нагрузки [7]. Эти выводы связаны с практикой внедрения архитектур ServerlessLLM, Torpor и близких систем, где механизмы отказоустойчивости строятся вокруг гибкого управления ресурсами и прозрачного восстановления функций.

В совокупности результаты анализа показывают, что устойчивые распределённые системы AI-инференса строятся на сочетании нескольких групп механизмов. На архитектурном уровне используются микросервисная декомпозиция, бессерверные и облачно-нативные паттерны [1], [2], [7], [9]. На уровне управления состоянием применяются многоуровневые чекпоинты, потоковая репликация KV-кэша и аккуратное разделение состояний по подсистемам [2], [6], [10]. На уровне деградации качества используются гетерогенная репликация и облегчённые модели-заменители [8]. На уровне данных и логики встраиваются механизмы мониторинга, диагностики и фильтрации некорректных входов и выходов [3], [5]. Такое сочетание механизмов формирует основу для проектирования инфраструктуры инференса, способной выдерживать отказы узлов, перегрузки, деградацию данных и нарушения в поведении моделей без длительной недоступности сервиса.

Обсуждение

Сопоставление рассмотренных систем позволяет провести разграничение между подходами, ориентированными на инфраструктурную устойчивость, и подходами, опирающимися на управляемую деградацию качества и интеллектуальное обнаружение аномалий. Облачно-нативные архитектуры [1], [7] опираются на стандартные механизмы кластерной отказоустойчивости: перезапуск контейнеров, перераспределение входящего трафика, проверку работоспособности сервиса (health check) и проверку готовности экземпляра к приёму запросов (readiness probe). ServerlessLLM и DéjàVu усиливают этот уровень, переходя от грубой репликации к более тонкому управлению состоянием моделей, использующему многоуровневое хранение чекпоинтов и потоковую передачу KV-кэша [2], [6]. FailLite и Torpor, в свою очередь, решают задачу сохранения работоспособности при крайне ограниченных ресурсах или нестабильной доступности GPU, предлагая гетерогенную репликацию и гибкое планирование нагрузки [8], [9].

Появление SGLang HiCache в open-source среде уточняет траекторию развития идей DéjàVu. Если в работе Strati и соавторов потоковая передача KV-кэша и репликация состояния описаны как средство быстрого восстановления и снижения накладных расходов LLM-сервинга, то HiCache переносит близкую логику в инженерный стек с иерархическим хранением, pluggable backends и настраиваемыми политиками записи [6], [11]. При такой постановке fault tolerance выступает как следствие аккуратной работы с промежуточным состоянием, когда потеря горячего кэша не ведёт к полному откату вычислений.

Для наглядного сравнения стратегий отказоустойчивости рассмотренных систем используется сводная таблица 1.

Таблица 1 - Сравнение стратегий отказоустойчивости в системах AI-инференса

DOI: https://doi.org/10.60797/itech.2026.11.7.2

Система	Тип среды	Основной механизм устойчивости	Особенности ресурсоиспользования
Cloud-Native архитектура	Облачный кластер	Репликация сервисов, перезапуск контейнеров, распределение входящего трафика.	Масштабирование экземпляров сервиса, применение сервисной сети.
ServerlessLLM	Инференс крупных языковых моделей в бессерверной среде	Многоуровневое хранение чекпоинтов, локальное при-GPU хранилище	Снижение времени холодного старта, уменьшение обращений к удалённому хранилищу
DéjàVu	Распределённый LLM-сервинг	Потоковая передача KV-кэша, раздельная обработка промпта и токенов, репликация состояния	Сокращение простоев стадий конвейерной обработки, экономия GPU-памяти
FailLite	Периферийный сервинг	Гетерогенная репликация, тёплые и холодные реплики, прогрессивное восстановление	Минимизация MTTR при ограниченных ресурсах, управляемая деградация качества
Torpor	Бессерверная платформа с поддержкой графических	Позднее связывание, перестановка моделей между GPU, интерференционно-	Высокая утилизация GPU, снижение затрат на выделение вычислительных

Система	Тип среды	Основной механизм устойчивости	Особенности ресурсоиспользования
	процессоров	учитывающий планировщик	ресурсов.
Hoplite / ExoFlow	Универсальные распределённые системы	Распределённое объектное хранилище, устойчивые коллективные операции, разделение единиц выполнения и восстановления	Снижение накладных расходов на восстановление, поддержка ровно-одного выполнения

Примечание: на основе источников [1], [2], [6], [8], [9], [10]

Дополнительный слой устойчивости формируется за счёт мониторинга и диагностики, как подчёркивают исследования A. Gogineni, L.S. Myllyaho и F.F.B. Santos [3], [4], [5]. Таблица 2 иллюстрирует связь между типами отказов и соответствующими механизмами обнаружения и реакции.

Таблица 2 - Типы отказов и методы обработки в системах AI-инференса

DOI: https://doi.org/10.60797/itech.2026.11.7.3

Тип отказа / нарушения	Источники описания	Примеры проявлений	Механизмы обработки и смягчения
Искажения входных данных	Облачные CV-сервисы [5]	Шум, низкое разрешение, пропуски фрагментов	Инъекция ошибок для оценки устойчивости, корректировка пайплайнов обработки данных
Некорректное поведение моделей	Архитектурные паттерны ML-систем [4]	Сдвиг распределения, аномальные выходы	Ограничения на входы и выходы, бизнес-правила, дублирование моделей разной сложности
Инфраструктурные сбои узлов и сети	Cloud-Native и бессерверные-системы [1], [2], [7], [9], [10]	Отказ узлов, сетевые задержки, перегрузка GPU	Репликация сервисов, балансировка, многоуровневые чекпоинты, перестановка моделей, устойчивые коллективные операции
Локальные аппаратные сбои на периферии	Периферийный сервинг FailLite [8]	Перегрев устройств, нестабильное питание, ограничение памяти	Гетерогенная репликация, тёплые и холодные реплики, прогрессивное восстановление
Комплексные сбои в рабочих процессах	Hoplite / ExoFlow [10]	Чередование ложных и реальных отказов в цепочке задач	Точное разделение путей выполнения и восстановления, семантика ровно-одного выполнения
Аномалии в распределённых системах	AI-управляемые механизмы отказоустойчивости [3]	Отклонения в телеметрии, комбинированные сбои	Применение моделей глубокого обучения, включая архитектуру VGG16, для раннего обнаружения отказов и классификации аномалий.

Примечание: на основе источников [1], [2], [3], [4], [5], [7], [8], [9], [10]

С точки зрения инженера серверных систем анализируемые работы предлагают несколько практических направлений. Во-первых, следует рассматривать управление состоянием модели (чекпоинты, KV-кэш, метаданные сессий) как отдельный слой, тесно связанный с оркестратором, но имеющий собственные механизмы резервирования [2], [6], [10]. Во-вторых, механизмы деградации качества, подобные гетерогенной репликации FailLite, целесообразно вводить там, где остановка сервиса недопустима, а небольшое снижение точности приемлемо [8]. В-третьих, мониторинг и диагностика, основанные на машинном обучении, служат дополнительным уровнем защиты, позволяя выявлять ситуации, в которых стандартные метрики инфраструктуры не дают полной картины [3], [5].

Наконец, требование обслуживания сотен AI-сервисов небольшой командой подводит к необходимости систематизировать эти приёмы в виде референсной архитектуры. Такой каркас включает облачно-нативную основу с оркестратором и сетями сервисов [1], [7], специализированные подсистемы сервинга LLM с учётом потоковой обработки KV-кэша [2], [6], отдельный модуль деградации качества для периферийных сценариев [8], GPU-ориентированный бессерверный слой [9] и универсальный слой устойчивой коммуникации и рабочих процессов [10]. Опора на подобный каркас упрощает внедрение новых моделей и сервисов без радикальной перестройки инфраструктуры и снижает нагрузку на эксплуатационную команду.

Большинство рассмотренных подходов применимо к сервисам без GPU и к системам, не связанным с LLM, поскольку автоматическое обнаружение аномалий, повышение доступности при изменении профиля нагрузки и восстановление после отказа части серверов значимы для широкого круга серверных систем. Различия между лёгкими CPU-сервисами и LLM-сервисами на GPU связаны прежде всего с длительностью восстановления, объёмом состояния модели и стоимостью повторной инициализации. Поэтому прямое перенесение описанных решений требует адаптации к вычислительному профилю сервиса, а смежные классы серверных приложений остаются за пределами данной статьи.

Заключение

Поставленные во введении задачи решены посредством аналитического обобщения современных подходов к отказоустойчивости в распределённых системах AI-инференса. Классифицированы механизмы устойчивости по уровням системы: архитектура сервисов, управление состоянием моделей, управление качеством и данные. Выделены архитектурные практики облачно-нативных и бессерверных систем, решения для LLM-сервинга и периферийных сценариев, а также методы контроля данных и поведения моделей, образующие многоуровневую защиту подсистем инференса.

Практическая состоятельность решений, основанных на тонком управлении KV-состоянием, подтверждается open-source реализациями. Публикация SGLang HiCache фиксирует переход подобных приёмов в рабочие стеки LLM-сервинга, где иерархическое кэширование и перенос состояния между уровнями памяти сочетаются с заметным сокращением времени до первого токена и ростом пропускной способности.

Сопоставление облачно-нативной архитектуры предиктивного моделирования, ServerlessLLM, DéjàVu, FailLite, Torpor и механизмов Hoplite / ExoFlow позволило выделить устойчивые инженерные приёмы: многоуровневое хранение чекпоинтов и KV-кэша, гетерогенную репликацию с управляемой деградацией качества, гибкое GPU-планирование, устойчивую коллективную коммуникацию и отделение путей выполнения и восстановления.

Формирование концептуальной референсной схемы инфраструктуры инференса, ориентированной на эксплуатацию большим числом моделей силами ограниченной команды объединяет облачно-нативный слой оркестрации, специализированные подсистемы сервинга LLM, модуль деградации качества для периферийных узлов, GPU-ориентированную бессерверную платформу и универсальную систему рабочих процессов. Такое объединение создаёт основу для практического проектирования надёжных серверных систем, где устойчивость к отказам достигается не за счёт единичного решения, а через согласованное взаимодействие нескольких уровней архитектуры.

Краткий глоссарий

AI-инференс — выполнение обученной модели на новых данных для получения результата: классификации, прогноза, распознавания, ответа языковой модели или иного машинного вывода.

LLM-сервинг — организация постоянной работы большой языковой модели как сетевого сервиса, принимающего запросы пользователей или приложений и возвращающего сгенерированный текст.

Оркестрация — автоматическое управление запуском, остановкой, переносом и масштабированием сервисов по серверным узлам, например в Kubernetes.

Репликация — создание и поддержание нескольких копий сервиса, модели или промежуточного состояния, чтобы при сбое одной копии система перенаправила запросы к другой.

KV-кэш — сохранённые промежуточные вычисления большой языковой модели в формате «ключ–значение», которые позволяют продолжить генерацию текста без полного повторного расчёта уже обработанного запроса.

Пул моделей или реплик — заранее выделенная группа экземпляров моделей, доступных для обработки запросов, резервного переключения или постепенного восстановления после сбоя.

Бессерверная среда — способ размещения вычислений, при котором разработчик задаёт функцию или сервис, а платформа сама выделяет ресурсы, запускает экземпляры под нагрузку и освобождает их после выполнения.

Конфликт интересов

Не указан.

Рецензия

Все статьи проходят рецензирование. Но рецензент или автор статьи предпочли не публиковать рецензию к этой статье в открытом доступе. Рецензия может быть предоставлена компетентным органам по запросу.

Conflict of Interest

None declared.

Review

All articles are peer-reviewed. But the reviewer or the author of the article chose not to publish a review of this article in the public domain. The review can be provided to the competent authorities upon request.

Список литературы на английском языке / References in English

1. Bajwa M.T.T. Cloud-Native Architectures for Large-Scale AI-Based Predictive Modeling / M.T.T. Bajwa, S. Wattoo, I. Mehmood [et al.] // Journal of Emerging Technology and Digital Transformation. — 2025. — Vol. 4. — № 2. — P. 207–221.

2. Fu Y. ServerlessLLM: Low-Latency Serverless Inference for Large Language Models / Y. Fu, L. Xue, Y. Huang [et al.] // Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2024). — 2024. — P. 1–19.

3. Gogineni A. Artificial Intelligence-Driven Fault Tolerance Mechanisms for Distributed Systems Using Deep Learning Model / A. Gogineni // Journal of Artificial Intelligence, Machine Learning and Data Science. — 2023. — Vol. 1. — № 4. — P. 2401–2406.

4. Myllyaho L.S. On Misbehaviour and Fault Tolerance in Machine Learning Systems / L.S. Myllyaho, M. Raatikainen, T. Männistö [et al.] // Journal of Systems and Software. — 2022. — Vol. 183. — DOI: 10.1016/j.jss.2021.111096.

5. Santos F.F.B. Assessing the Reliability of Machine Learning Cloud Services Through Fault Injection / F.F.B. Santos. — Universidade Federal de Alagoas, Instituto de Computação, 2022. — 47 p.

6. Strati F. DéjàVu: KV-Cache Streaming for Fast, Fault-Tolerant Generative LLM Serving / F. Strati, S. McAllister, A. Phanishayee [et al.] // Proceedings of the 41st International Conference on Machine Learning (ICML 2024). — 2024. — Vol. 235. — P. 1–27.

7. Wang L. Advancing Serverless Computing for Scalable AI Model Inference: Challenges and Opportunities / L. Wang, Y. Jiang, N. Mi // Proceedings of the 10th International Workshop on Serverless Computing (WoSC 10). — 2024. — P. 1–6.

8. Wu L. FailLite: Failure-Resilient Model Serving for Resource-Constrained Edge Environments / L. Wu, W.A. Hanafy, T. Abdelzaher [et al.] // arXiv. — 2025. — URL: https://arxiv.org/abs/2504.15856 (accessed: 10.04.2026).

9. Yu M. Torpor: GPU-Enabled Serverless Computing for Low-Latency, Resource-Efficient Inference / M. Yu, A. Wang, D. Chen [et al.] // Proceedings of the USENIX Annual Technical Conference (ATC). — 2025. — P. 1–18.

10. Zhuang S. Providing Efficient Fault Tolerance in Distributed Systems / S. Zhuang. — Berkeley, CA : University of California, Berkeley. Technical Report UCB/EECS-2024-86. — 2024. — URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/Archive/EECS-2024-86.pdf (accessed: 10.04.2026).

11. Xie Z. SGLang HiCache: Fast Hierarchical KV Caching with Your Favorite Storage Backends / Z. Xie // LMSYS Blog. — 2025. — URL: https://www.lmsys.org/blog/2025-09-10-sglang-hicache/ (accessed: 10.04.2026).