

МОДЕЛИРОВАНИЕ УСТОЙЧИВОСТИ АРХИТЕКТУР В УСЛОВИЯХ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ДАННЫХ

Демэйн Э., Computer Visions Master's Degree Programm, II курс
Открытый университет Каталонии, Барселона, Испания

Моделирование устойчивости архитектур в условиях распределенной обработки данных опирается на представление системы как совокупности взаимодействующих узлов вычисления, коммуникации и хранения, где поведение определяется не только алгоритмами, но и физикой сети, политиками консистентности и неоднородностью потоков данных. Устойчивость здесь означает предсказуемость результата и деградацию «по грации» при отказах, задержках, дрейфе распределений, стрессе нагрузки и ошибках локальных компонентов. В отличие от централизованных схем, распределённая среда добавляет асинхронность, частичную наблюдаемость и необходимость выбора между консистентностью, доступностью и задержкой. Поэтому модель устойчивости должна одновременно учитывать вероятностную природу событий в сети, вариативность данных и алгоритмическую чувствительность архитектуры — будь то тренировка больших моделей, потоковая аналитика, RAG-конвейеры или федеративное обучение [1].

Отправной точкой является явное разделение источников риска. Коммуникационные сбои приводят к задержкам, потерям и дублированию сообщений; вычислительные узлы дают ошибки времени выполнения, «зависания» и неоднородную производительность; хранилища создают дивергенции версий и эффект «грязных» чтений; данные порождают дрейф доменов, разные шкалы и пустые зоны покрытий. Для каждого класса вводится свой контур тестов: для сети — инъекции задержек и потерь, переменные пропускные способности, сетевые разрезы; для вычислений — инъекции «медленных» задач и спорадических падений; для данных — перепутанные единицы, контрфактуальные сдвиги, перекосы частот. Смысл моделирования — не построить идеальный симулятор мира, а создать репертуар воспроизводимых сцен, в которых архитектура проявляет латентные хрупкости до продакшна.

Архитектуру имеет смысл описывать через уровни: транспорт и RPC, слой координации и консенсуса, слой хранения и индексов, слой вычислений/планировщика, слой алгоритмов и прикладных политик. На транспортном уровне устойчивость — это гарантии доставки с bounded retry и backoff, дубликация и идемпотентность обработчиков. На уровне координации — выбор между строгой консистентностью и eventual consistency; многие задачи допускают политику «чтение из локального, запись через журнал», если прикладной

слой умеет работать с версионностью. На уровне хранения критично вводить атомарные границы и схемы версионирования: снабжать документы лампорт- или векторными часами, хранить компактные «истины» в виде квазинормализованных фактов и агрегатов, а не «сырого» текста. На уровне вычислений важно следить за утилизацией, очередями и «хвостами задержек», которые чаще всего и убивают SLO. Наконец, алгоритмический слой должен иметь свои защитные бортики: пороги отказов «не знаю», ретривал по белым спискам, проверка единиц, явное ветвление ответов при конфликте источников.

В задачах обучения и инференса больших моделей основная угроза — асинхронность и стейлнес обновлений. В параметр-серверных и Totally Asynchronous схемах градиенты приходят с задержкой Δ , а их применение создаёт дрожание качества. Практический компромисс — частично синхронные окна, буферизация и клиппинг экстремальных лагов; на больших кластерах помогает «быстрая кворумная синхронизация» с адаптивным порогом: достаточно q из n рабочих, где q динамически регулируется нагрузкой и целевой вариативностью метрик. Ускорение передачи достигается компрессией и квантизацией градиентов, но устойчивость требует компенсирующих схем: случайной ротации знаков, коррекции смещения, периодических полноточных «якорей» и согласования масштабов слоёв через RMS- или LayerNorm на стороне обучающих рабочих. Для алгоритмической устойчивости полезны плоскостные процедуры (EMA/SWA, SAM/ASAM) — они уменьшают остроту минимума и, как следствие, чувствительность к несинхронным обновлениям.

В федеративном сценарии ключевой риск — не IID-распределения и нестабильная коммуникация. Чтобы «модель большинства» не ломала редкие домены, применяют важностные веса клиентов, стратификацию по кластерам данных и доверительным меткам, а также механизмы отбора/переотбора клиентов на раунде. С точки зрения устойчивости агрегации безопаснее медианные и триммированные правила вместо простого среднего: они гасят выбросы и сдерживают потенциальные отравления. Дифференциальная приватность добавляет шум, и здесь уместна явная «бухгалтерия полезности»: мониторинг метрик до/после шума с адаптивным бюджетом. При неустойчивых сетях федеративная схема должна уметь «замыкается» локально: выполнять ограниченную адаптацию в офлайне и синхронизироваться при возможности, сохраняя неизменными ключевые инварианты интерфейса [2].

В конвейерах RAG уязвимость часто кроется в распределённом ретривере и кэше. Моделирование включает «грязные» индексы, устаревшие сегменты, перекосы доменов и резкие скачки релевантности. Устойчивое решение — двухконтурный поиск: быстрый, но шумный кандидатный и медленный rerank по полезности и доверию; кэш должен быть версионным, с TTL и аннотированными источниками. Для борьбы с сетевыми «провалами» ретривер работает в режимах деградации: при высоких задержках — меньший k , строгие доменные фильтры, принудительное цитирование; при отсутствии индекса —

откровенный отказ или ответ-шаблон с инструкциями, но не «красивая выдумка». Децентрализованные кэши требуют стратегий согласования: слабая консистентность допустима, когда прикладной слой умеет показывать версию и свежесть, а критичное — извлекается только из «жёсткого» контура.

Для потоковой аналитики и онлайн-моделей узкое место — порядок и полнота событий. В симуляции полезно воспроизводить задержки по воде, out-of-order, дубликаты, редкие «бурсты». Устойчивые архитектуры фиксируют семантику времени (processing vs event time), используют водоразделы, идемпотентные операторы, «поглощающие» агрегаты и восстановление из снимков. Бэкпрешер — ещё один скрытый убийца когерентности: без регуляции очередей система сначала начинает задерживать низкоприоритетные потоки, затем теряет события, затем деградируют модели. Правильный ответ — управляющие контуры с целевыми задержками, адаптивными частотами контрольных точек и каналами «быстрой полосы» для управляющих сигналов.

Хорошее моделирование устойчивости немыслимо без инжиниринга отказов. Инъекции «страгглеров», выключения узлов, сетевые разрезы, сбои в координации, частичные откаты хранилищ, повреждённые блоки индекса — всё это должно быть частью автоматизированной тестовой матрицы. Полезно задавать профили интенсивности ϵ и длительности τ : короткие, но частые микросбои; редкие, но тяжёлые разрывы; ступенчатое ухудшение канала. Система считается устойчивой, если ключевые сервисные метрики (доля успешных ответов, медиана и 95-й перцентиль задержки, доля отклонённых запросов, доля условных/отказных ответов) остаются в коридорах при росте ϵ , τ , а деградация носит контролируемый и обратимый характер.

Важный компонент — модели наблюдаемости. На инфраструктурном уровне собираются латентности RPC, процент повторных попыток, процент дедупликаций, глубина очередей, кворумы и пересчёты выборов лидера. На уровне данных — дивергенции распределений, «дрейф» признаков, частоты редких токенов и триггеров, доля конфликтов источников в RAG. На алгоритмическом уровне — межзапусковая дисперсия энтропии токенов, «логит-дрейф» на якорных запросах, частота противоречий и нестабильных отказов. Эти панели не просто визуализируются: поверх них строятся EWMA/CUSUM карты и политики эскалаций. Выход из коридора — автоматическое охлаждение декодера, усиление rerank, повышение порога отказа, ограничение длины ответов, включение строгих шаблонов, канареечное откатывание на предшествующий профиль [3].

Архитектуры с экспертными слоями (MoE) добавляют собственную механику сбоев: маршрутизация по экспертам может быть ступенчатой и чувствительной к мелким сдвигам токенизации; при сетевых проблемах часть экспертов недоступна. Смягчение достигается энтропийной регуляризацией гейта, каппингом нагрузки и резервированием критичных экспертов; при недоступности — деградация к «универсальному» эксперту с объяснимым

снижением качества, но без «ломки» смысла. Планировщик должен понимать распределение нагрузки и уметь преэмптить горячие узлы, не допуская «эффекта лифта», когда задержка одного эксперта срывает последовательность целиком.

Моделирование устойчивости включает и экономику вычислений. Под «бурсты» трафика нужна эластичность: горизонтальный автоскейлинг по очередям, а не только по CPU; приоритеты для критичных запросов; «отсечка» генерации по времени с сохранением когерентности. В тренировке действенна «многочастотность»: быстрые контроли для явных сбоев качества и медленные, но точные — для сдвигов по доменам. При дефиците пропускной способности сети логика маршрутизации должна минимизировать межзонные передачи: локальные градиенты, локальный ретривал, локальные кэши — и только агрегаты уходят «в центр».

Ключ к практической применимости — протоколы сертификации. Их задача — доказать, что при заданном профиле отказов и дрейфов архитектура остаётся в допусках. Входами служат матрица сценариев, коридоры метрик и правила деградации. Достаточно часто полезна вероятностная формулировка: при интенсивности событий λ и доле злонамеренных входов ρ вероятность выхода метрик за пределы меньше δ . Это не математическая строгость в чистом виде, а операционная гарантия с привязкой к процессам: если наблюдаемый поток пересекает плановый λ или ρ , включается консервативный профиль, а не надеется «пронесёт».

Наконец, распределённая обработка — это неизбежные компромиссы. Некоторые системы должны выбирать доступность и низкую задержку, принимая eventual consistency, и компенсировать это прикладным уровнем: версионированием и условными ответами. Другие обязаны держать строгую согласованность и мириться с отказами при разрезах. В первых важны форматы ответа, ясные отметки «при условии версии v », и «страховые» проверки; во-вторых — резервирование, быстрые выборы лидера и предсказуемость таймаутов. В обоих случаях устойчивость — это не одна хитрая эвристика, а связка: чистые входы и глоссарии, архитектурная дисциплина подсказок и ретривала, адаптивный декодер с профилями деградации, верификация смысла, наблюдаемость с автоматическими вмешательствами и регулярные инъекции сбоев. Когда все эти слои работают согласованно, распределённая архитектура перестаёт быть «организмом с сюрпризами» и превращается в систему, чья семантика и производительность остаются контролируемыми даже в мире с задержками, частичными отказами и изменчивыми данными [4].

Список литературы:

1. Li, B. Gradient purification: defense against data poisoning attack in decentralized federated learning / B. Li, X. Miao, Y. Zhang [et al.] // Frontiers of

-
- Computer Science. – 2026. – Vol. 20, no. 6. – Art. 206820. – DOI 10.1007/s11704-025-50240-3.
2. Thamilarasu, G. SpaceTime: A Deep Similarity Defense Against Poisoning Attacks in Federated Learning / G. Thamilarasu, C. Dunham // Big Data and Cognitive Computing. – 2025. – Vol. 9, no. 12. – Art. 313. – DOI 10.3390/bdcc9120313.
 3. Ren, Q. Shadow backdoor attack: Multi-intensity backdoor attack against federated learning / Q. Ren, Y. Zheng, C. Yang [et al.] // Computers & Security. – 2024. – Vol. 139. – Art. 103740. – DOI 10.1016/j.cose.2024.103740.
 4. Yu, F. Chronic Poisoning: Backdoor Attack against Split Learning / F. Yu, B. Zeng, K. Zhao [et al.] // Proceedings of the AAAI Conference on Artificial Intelligence. – 2024. – Vol. 38, no. 15. – P. 16531–16538. – DOI 10.1609/aaai.v38i15.29591.