Копыркин Николай Андреевич
аспирант
АНО ВО «Российский новый университет»
г. Москва
Судакова Анна Дмитриевна
магистрант
ФГБОУ ВО «Московский государственный
технический университет им. Н.Э. Баумана»
г. Москва

DOI 10.31483/r-167403

АППАРАТНЫЕ АРХИТЕКТУРЫ ДЛЯ ЛОКАЛЬНОГО ОБУЧЕНИЯ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ: ОБЗОР МАСШТАБИРУЕМЫХ РЕШЕНИЙ

Аннотация: *в статье рассматривается выбор аппаратных ускорителей (GPU, ASIC, FPGA и др.) для обучения больших языковых моделей на собственных серверах (on-premise), а не в облаке. Проводится сравнительный анализ стоимости владения различными решениями при установке 256 ускорителей. Делается вывод, что выбор оборудования должен учитывать не только скорость вычислений, но и затраты на электроэнергию, охлаждение и программную совместимость.*

Ключевые слова: *большие языковые модели, аппаратные ускорители, масштабируемые архитектуры, совокупная стоимость владения, локальное развёртывание.*

В последние годы модели искусственного интеллекта стали намного сложнее: например, современные большие языковые модели (LLM) имеют сотни миллиардов параметров и требуют квинтиллионы операций. Из-за этого без специального оборудования учить их на обычных процессорах (CPU) просто не получится [1].

Обучение одной модели использует мегаватт-часы электричества, поэтому энергоэффективность – это не опция, а необходимость. А если сервера размещены локально, то система ещё должна нормально масштабироваться: добавление ресурсов даёт пропорциональный прирост производительности.

Цель этой работы – разобраться, какие аппаратные архитектуры подходят для обучения ИИ на собственных серверах, как они масштабируются и во сколько реально обходятся. Для этого были изучены публикации и документация производителей за 2022–2026 годы. В обзоре рассматриваются GPU, ASIC и FPGA. Главный критерий – чтобы мощности росли, а расходы на оборудование, электричество и обслуживание оставались под контролем.

Разберём основные наиболее распространённые архитектуры ускорителей. Рассмотрим их преимущества и недостатки.

Универсальные ускорители (GPU, TPU, IPU).

В инфраструктурах для AI в большинстве случаев применяются массово-параллельные процессоры. Изначально эти устройства были графическими, но позже их адаптировали, чтобы они выполняли тензорные операции. Данные чипы занимают основную долю рынка в процессах обучения и инференса. Эти системы легко адаптировать под разные задачи, и они показывают максимально возможную скорость вычислений. Однако устройства имеют меньшую энергоэффективность в сравнении с ASIC и зависят от пропускной способности памяти. Требуют высоких капитальных затрат.

Примеры: NVIDIA H100/B200, AMD MI350X, Google TPU v7e.

Специализированные ASIC для вывода.

ASIC (Application-Specific Integrated Circuit) – это устройства, спроектированные на выполнение только строго определенных действий, таких как свёртки, умножение или работы механизма внимания. Благодаря этому они не расходуют свою производительность на посторонние задачи и способны достичь высокой скорости работы при малых затратах энергии. ASIC хорошо подходят для больших систем, таких как облачные программные интерфейсы и алгоритмы рекомендаций. Они эффективно расходуют энергию на каждую единицу вычислений

и имеют низкую общую стоимость владения. Данные чипы невозможно перенастроить под другие задачи или использовать для обучения, а также они привязаны к вендору.

Примеры: AWS Inferentia2, Google Edge TPU.

Экспериментальные архитектуры (FPGA, PIM, фотонные).

К данному виду архитектур относятся: PIM (Processing-In-Memory) – вычисления в памяти, фотонные ускорители и FPGA (Field-Programmable Gate Arrays) и другие более экспериментальные архитектуры. Вычисления в памяти сокращают перемещение данных и энергию, а фотонные ускорители используют свет для умножения матриц, обеспечивая высокую пропускную способность. Они пока не достигли массового распространения, потому что находятся в стадии активных исследований. FPGA же занимают место между специализированными ASIC и универсальными процессорами. Они позволяют быстро адаптировать аппаратную архитектуру под конкретную модель или даже под конкретный слой нейронной сети [2].

FPGA имеют хорошую гибкость (можно быстро адаптировать под новые алгоритмы), низкую латентность и хорошее соотношение производительности на ватт. PIM позволяет снизить потребление энергии на 40–90%, а фотонные ускорители имеют огромную пропускную способность. Архитектуры отличаются технологической незрелостью, сложностью интеграции и отсутствием стандартного ПО.

Примеры: Xilinx Alveo (FPGA), HPIM/P3-LLM (PIM).

Гетерогенные системы.

Различные типы вычислительных устройств, такие как CPU, GPU, NPU и FPGA, работают вместе в одной среде. В этой системе они используют общее адресное пространство и соединяются через каналы с высокой пропускной способностью, например NVLink, NeuronLink-v4 или Infinity Fabric. По этой причине задачи распределяются между компонентами в зависимости от их функций. Для управления процессами применяются CPU, а для выполнения объемных операций с тензорами используются GPU или NPU. На таких принципах строятся

суперкомпьютеры и крупные AI-кластеры. Но создание программ для таких систем является трудоемким процессом и требует наличия специализированных фреймворков.

Благодаря этому задачи распределяются в соответствии с возможностями устройств, при этом систему можно расширять, и она работает продуктивно при разных типах нагрузки. Однако межсоединения имеют высокую стоимость, процесс написания программного обеспечения является сложным, и возникает зависимость от конкретного производителя.

Примеры: NVIDIA Grace Hopper, Fujitsu A64FX, Apple M-серии.

Сравнительная таблица ключевых ускорителей.

Чтобы сопоставить, как разные архитектуры выполняют задачи по обучению и инференсу LLM, применяется система параметров. В эту систему входят показатели, описывающие наибольшую возможную скорость вычислений в TFLOPS или PFLOPS, количество доступной памяти и скорость передачи данных в ней. Также учитываются количество потребляемой энергии в ваттах (TDP) и способность соединений объединять устройства в крупные кластеры. В Таблице 1 указаны технические характеристики наиболее распространённых AI-ускорителей с различными архитектурами. Для формирования этих данных используются сведения от компаний-изготовителей и результаты тестов производительности за период с 2024 по 2026 год [3–6].

Таблица 1

Сравнительные характеристики AI-ускорителей для LLM

Характеристика	NVIDIA H100	NVIDIA B200	AMD MI350X	Google TPU v7e Ironwood	AWS Trainium3	Intel Gaudi 3
Техпроцесс, nm.	4	4	3	н/д	3	5
Тип	GPU	GPU	GPU	TPU	ASIC	NPU
Транзисторы, млрд.	80	104	185	н/д	н/д	н/д
FP8, PFLOPS	4	9	9.2	4.6	2.52/чип	1,8
FP4, PFLOPS	н/д	18	18.4	н/д	н/д	н/д
Память (HBM), GB	80 HBM3	192 HBM3e	288 HBM3e	192	144 HBM3e	128 HBM2e

Пропускная способность, TB/s	2.04	8	8	7.37	4.9	3.7
TDP (Ватт)	350	1000	1000	980	500	900
Межсоединение TB/s	0.9 NVLink	1.8 NVLink	1 Infinity Fabric	1.2 Inter-Chip Interconnect	2 NeuronLink-v4	0.6 RDMA

Сравнительный анализ совокупной стоимости владения (TCO).

При сравнении TCO для AI-инфраструктуры специалисты оценивают капитальные затраты (CAPEX) при покупке оборудования и операционные расходы (OPEX) при оплате электроэнергии, охлаждения, обслуживания и программного обеспечения. В Таблице 2 указаны примерные расчетные значения для кластера из 256 ускорителей.

Таблица 2

Сравнение TCO для кластера из 256 ускорителей

Показатель	NVIDIA H100	NVIDIA B200	AMD MI350X	Google TPU v7e	AWS Trainium3	Intel Gaudi 3
CAPEX, млн $.	8–10	15–20	10–12	12–15	6–8	6–8
Годовой OPEX, млн $.	1,2–1.5	2,0–2.5	1,5–1,8	1,0–1,3	0,9–1,1	0,8–1,0
TCO за 3 года, млн $.	11–14	21–27	14–17	15–19	8–11	8–11
TCO на 3 года на PFLOPS (тыс. $/PFLOPS)	10,7–13.6	9,1–11,7	5,9–7,2	12,7–16,1	12,4–17	17–23,4

Выводы по сравнительному анализу.

1. Универсальные GPU подходят для сбалансированной работы с существующими моделями и программными средами, что важно для организаций, если те не планируют изменять программное обеспечение. Но высокая стоимость владения является приемлемой только тогда, когда пользователи обучают модели, имеющие более 100 млрд параметров. Если компания ориентируется на открытые программные среды, то стоит выбрать решения от AMD.

2. Специализированные ASIC и TPU используются преимущественно в крупных проектах. Ускоритель от компании Google имеет низкое потребление, но это оборудование функционирует только внутри Google Cloud. AWS

Trainium3 требует самых малых затрат на покупку и имеет минимальный TCO среди всех вариантов. Однако его производительность скромнее в сравнении с конкурентами. Подобные решения являются подходящими для крупных облачных провайдеров и компаний, которые уже используют соответствующие программные среды.

3. Для малых и средних нагрузок, где модели имеют менее 70 миллиардов параметров, подходит Intel Gaudi 3. Использование этого оборудования требует низких затрат и обеспечивает достаточную скорость вычислений.

Исходя из Таблицы 2, наименьшую удельную стоимость демонстрирует AMD MI350X, за ним следуют NVIDIA B200 и H100. В этой же таблице Intel Gaudi 3 имеет самую высокую цену по данному показателю. Но малый объём начальных вложений в Gaudi 3 делает его подходящим для первых этапов работы или для задач, где не требуется высокая плотность вычислений.

Подбор аппаратной архитектуры для обучения ИИ на своих мощностях зависит от многих факторов. Организациям следует выполнять тесты своих моделей через бенчмарки. Это поможет принять решение, которое учитывает фактические нагрузки и будущие траты на эксплуатацию.

Вызовы и будущие направления.

Несмотря на достижения в аппаратном обеспечении для ИИ, до полного решения всех проблем ещё далеко. Когда такие системы начинают внедрять в реальные проекты, сразу появляется множество ограничений, которые простым наращиванием вычислительных мощностей не обойти.

Многие из этих ограничений имеют фундаментальную природу – они упираются в физику, экономику или сложившиеся экосистемы. Кое-какие вопросы удалось закрыть хотя бы частично с помощью оптимизации. Но другие по-прежнему остаются открытыми и ждут своих исследователей. Ниже – ключевые вызовы для тех, кто разворачивает обучение ИИ на собственных серверах.

1. Замедление закона Мура. Уменьшение технологических норм (переход на более тонкие техпроцессы) уже не даёт прежнего выигрыша ни по быстродействию, ни по энергоэффективности. Иными словами, простого масштабирования

транзисторов становится недостаточно. На первый план выходят уже не столько технологические ухищрения, сколько архитектурные решения – пересмотр самой организации вычислений.

2. «Стена памяти». Пропускная способность памяти увеличивается более медленными темпами, чем производительность процессоров. Это влияет на фактическую производительность обучения и инференса LLM из-за роста количества кластеров.

3. Расход энергии и системы охлаждения. Для обучения больших моделей требуются мегаватты электроэнергии. Также охлаждение большего количества тепла становится всё труднее в каждом последующем техпроцессе.

4. Программная привязка (vendor lock-in). Преобладание экосистемы компании NVIDIA создаёт значительное препятствие перед другими архитектурами. Это затрудняет переход на альтернативные архитектуры, увеличивает затраты на разработку ПО и ведёт к росту TCO.

5. Отсутствие общеизвестных метрик TCO. Сравнение архитектур по пиковой производительности (TFLOPS) не совсем корректно. Возможно, в будущем потребуются более стандартизированные методики оценки совокупной стоимости владения с учетом энергопотребления, охлаждения, обслуживания и ПО.

Будущие направления.

1. Вычисления внутри памяти (PIM). Сейчас данные передаются между памятью и процессором, и это увеличивает время работы и энергопотребление. Если встроить вычисления прямо в HBM или SRAM, этот лишний путь исчезнет – и задержки упадут, и энергии уйдёт меньше.

2. Единые стандарты межсоединений. Сегодня чиплеты от разных производителей плохо работают вместе – каждый вендор продвигает лишь свои решения. Открытые стандарты это исправят: можно будет собирать систему из компонентов разных компаний, как конструктор, не завися от одного поставщика.

3. Открытые программные экосистемы. CUDA от NVIDIA лидирует на рынке, и перейти на другое оборудование без потерь сложно. Инструменты типа MLIR, Triton и OneAPI постепенно меняют эту ситуацию – они не привязаны к

конкретному производителю и делают код более переносимым между архитектурами.

4. ИИ на службе у разработчиков железа. Проектирование чипов – долгий и дорогой процесс. Но если подключить LLM к генерации архитектур и поиску ошибок, цикл разработки можно заметно сократить: инженер задаёт требования, модель предлагает варианты, человек проверяет и правит.

Заключение.

В этом обзоре были рассмотрены современные аппаратные архитектуры, которые ориентированы на обучение моделей ИИ в условиях локального развёртывания (on-premises). Акцент делался на двух вещах – масштабируемости и совокупной стоимости владения.

GPU долгое время были главным выбором, но сложность современных LLM растёт быстрее, чем успевают за ней универсальные ускорители. Поэтому всё больше внимания получают специализированные решения – они выдают гораздо больше вычислений на каждый потреблённый ватт.

Если разворачивать инфраструктуру на собственных мощностях, а не в облаке, смотреть только на пиковую производительность – ошибка. Электричество, охлаждение, обслуживание и то, насколько хорошо ускорители работают с ПО – всё это в сумме может перевесить любые цифры из спецификации.

При этом ряд вопросов до сих пор открыт. Закон Мура замедляется, и простым уменьшением техпроцесса уже не обойтись – нужны новые архитектурные идеи. Отвод тепла становится серьёзным ограничением по мере роста плотности вычислений. А единых метрик для честного сравнения TCO по-прежнему нет, что сильно осложняет выбор между решениями.

Поэтому при проектировании или выборе аппаратной инфраструктуры для обучения ИИ на локальных серверах важно придерживаться системного подхода. Подбирать аппаратуру для локального обучения нужно комплексно – смотреть на масштабируемость, энергоэффективность, совместимость с используемым стеком и на то, в какую сумму обойдутся годы непрерывной эксплуатации.

References

1. Fanlong Z. Distributed training of large language models: A survey / Z. Fanlong, G. Wensheng, W. Yongheng, S.Y. Philip // Natural Language Processing Journal. – 2025. – No. 12. – P. 1–20.

2. FPGA Co-Design for Efficient N:M Sparse and Quantized Model Inference / F.-Y. Hsieh, Y.-C. Teng, D.-Y. Hong, J.-J. Wu // arXiv.org. – URL: https://arxiv.org/abs/2512.24713v1 (date of access: 05.04.2026).

3. Choquette J. NVIDIA Hopper H100 GPU: Scaling Performance / J. Choquette // IEEE Micro. – Вашингтон: IEEE Computer Society, 2023. – P. 9–17.

4. NVIDIA Hopper Architecture In-Depth / M. Andersch, G. Palmer, R. Krashinsky [et al.] // developer.nvidia.com. – URL: https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/ (date of access: 13.04.2026).

5. Jouppi N.P. Ironwood: Delivering Best in Class perf, perf/TCO and perf/Watt for Reasoning Model Training and Serving / N.P. Jouppi, S. Lakshmanamurthy // 2025 IEEE Hot Chips 37 Symposium. – URL: https://hc2025.hotchips.org/assets/program/conference/day2/61_Google_Ironwood-Final.pdf (date of access: 13.04.2026).