

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

**федеральное государственное бюджетное образовательное учреждение
высшего образования**

«Нижегородский государственный технический университет им. Р.Е. Алексеева»

Факультет подготовки специалистов высшей квалификации

Кафедра «Прикладная математика»

ОТЧЕТ

о педагогической практике

по направлению подготовки кадров высшей квалификации
09.06.01 «Информатика и вычислительная техника»
направленности «Системный анализ, управление и обработка информации»

Заведующий кафедрой,
д.ф.-м.н., профессор



30.05.2025

/Пакшин П.В./

(подпись, дата)

Научный руководитель,
д.ф.-м.н., профессор



30.05.2025

/Пакшин П.В./

(подпись, дата)

Исполнитель
аспирант



/Комаров А.О./

(подпись, дата)

Нижний Новгород 2025

**Индивидуальный план аспиранта
по педагогической практике**

Комаров Александр Олегович
(ФИО)

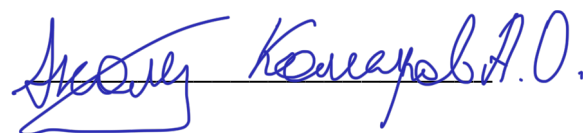
№	Содержание разделов работы; основные виды деятельности	Сроки выполнения	Отметка о выполнении
1.	Ознакомление с образовательной деятельностью организации, организацией учебного процесса и регламентом работы кафедры	01.03.2025 – 10.03.2025	
2.	Ознакомление с рабочей программой избранной учебной дисциплины	01.03.2025 – 10.03.2025	
3.	Работа с литературой на тему «Логика и архитектура вычислительных сред»	13.03.2025 – 24.03.2025	
4.	Анализ и обобщение полученной информации	27.03.2025 – 07.04.2025	
5.	Составление раздела учебно-методической документации специально курса дисциплины «Логика и архитектура вычислительных сред»	10.04.2025 – 19.05.2025	
6.	Оформление отчета о педагогической практике	22.05.2025 – 26.05.2025	

Подпись руководителя программы практики



/ Пакшин П.В. /

Подпись аспиранта



Отчет аспиранта о педагогической практике

1. Прделанная работа. Проведено ознакомление с организацией учебного процесса и регламентом работы кафедры «Прикладная математика», рабочей программой учебной дисциплины «Логика и архитектура вычислительных сред». Изучена литература на тему «Архитектуры вычислительных сред» с последующим анализом и обобщением полученной информации. Составлен раздел учебно-методической документации специально курса дисциплины «Логика вычислительных сред», который включает в себя семь основных разделов: цель и задачи дисциплины, место дисциплины в учебном процессе, требования к результатам освоения дисциплины, содержание дисциплины (базовые концепции и архитектуры вычислительных сред, аналитические хранилища данных и Big Data, искусственный интеллект и нейронные сети, автономные системы и робототехника, современные вычислительные среды для AI и робототехники), образовательные технологии, учебно-методическое обеспечение дисциплины, материально-техническая база дисциплины.

2. Соответствие индивидуальному плану. За время прохождения педагогической практики индивидуальный план выполнен в полном объеме.

3. Самооценка по проделанной работе (трудности, соответствие ожиданиям, успехи). Считаю, что с поставленными задачами (приобретение практических навыков и компетенций профессиональной педагогической деятельности, изучение основ учебно-методической и педагогической деятельности, ознакомление с современными результатами теории управления) справился успешно. Умения, навыки, знания и бесценный опыт, приобретенные мной при прохождении педагогической практики, несомненно, пригодятся в дальнейшей научно-исследовательской деятельности.

4. Предложения по проведению практики. Предложений по совершенствованию содержания и организации практики не имеется.

Подпись руководителя программы практики



/ Пакшин П.В. /

Подпись аспиранта

Раздел учебно-методической документации ~~специально-курса~~ дисциплины «Логика и архитектура вычислительных сред»

1. Цель и задачи дисциплины

1.1. Цель дисциплины

Формирование у студентов системного понимания принципов организации вычислительных сред, применяемых в современных технологиях обработки больших данных, искусственного интеллекта и автономных систем. Курс направлен на изучение архитектурных решений, обеспечивающих эффективную работу аналитических хранилищ данных (АХД), нейросетевых моделей, роботизированных комплексов и систем реального времени.

1.2. Задачи дисциплины

1.2.1 Теоретические аспекты

1. Изучение базовых принципов построения вычислительных систем (фон Неймана, параллельные и распределённые архитектуры).
2. Анализ современных процессорных технологий (CPU, GPU, TPU) и их роли в AI/Big Data.
3. Big Data и аналитические хранилища (АХД).
4. Освоение технологий хранения и обработки больших данных (Hadoop, Spark, колоночные СУБД).
5. Изучение методов оптимизации запросов и работы с Data Lakes.
6. Искусственный интеллект и нейронные сети:
 - исследование архитектур нейросетей (CNN, RNN, Transformer) для задач прогнозирования (погода, финансы, поведенческие модели);
 - практическое применение ML/DL в распределённых средах (обучение, инференс, масштабирование);
 - автономные системы и робототехника;

- принципы работы автопилотов (компьютерное зрение, LIDAR, управление в реальном времени);
- анализ взаимодействия сенсоров и систем управления (дроны, беспилотные автомобили).

7. Современные вычислительные платформы:

- облачные и edge-решения (AWS, NVIDIA Jetson) для AI и IoT;
- вопросы безопасности, latency и энергоэффективности.

1.2.2. Практическая подготовка

- Разработка проектов на стыке AI, Big Data и робототехники.
- Работа с инструментами (TensorFlow, PyTorch, ROS, Apache Kafka).

Итоговый навык: Способность проектировать и оптимизировать вычислительные среды для сложных задач, включая AI-аналитику, автономные системы и обработку данных в реальном времени.

2. Место дисциплины в учебном процессе

2.1. Статус дисциплины

Дисциплина «Логика и архитектура вычислительных сред» относится к обязательной части образовательной программы высшего образования АПИ НГТУ им. Алексеева по направлению подготовки 01.04.04 «Прикладная информатика».

2.1. Базовые дисциплины

Для успешного освоения курса студенты должны обладать знаниями из следующих дисциплин: «Архитектура ЭВМ и операционные системы» (принципы работы процессоров, памяти, ввода-вывода), «Базы данных и распределенные системы» (основы SQL, NoSQL, репликации, шардинга), «Программирование и алгоритмы» (Python/C++, основы параллельных вычислений), «Основы искусственного интеллекта» (машинное обучение, нейронные сети).

2.2. Последующие дисциплины

Результаты обучения, полученные в рамках дисциплины, необходимы для изучения следующих курсов: «Параллельное и распределенное программирование», «Средства разработки современного программного обеспечения», «Современная теория управления», «Нечеткие модели», «Защита информации».

Также знания курса критически важны при выполнении выпускной квалификационной работы, связанной с AI, Big Data или автономными системами.

2.3. Индивидуальный подход

Рабочая программа дисциплины разрабатывается индивидуально с учетом:

- технических возможностей (альтернативные форматы лекций, лабораторных);
- специализированного ПО (голосовые ассистенты).

2.4. Практическая интеграция

Курс напрямую связан с проектной деятельностью в рамках учебного плана, включая:

- разработку MVP для AI-моделей (предсказание, NLP, CV);
- развертывание облачных хранилищ данных (ClickHouse);
- симуляцию робототехнических систем (Gazebo, симулятор ВАЗ).

Вывод: Дисциплина формирует ключевые компетенции для работы с современными вычислительными средами, обеспечивая связь между фундаментальными знаниями (архитектура ЭВМ) и прикладными технологиями (AI, Big Data, робототехника).

3. Требования к результатам освоения дисциплины

3.1. Овладеть теоретическими знаниями

3.1.1. Основные принципы организации современных вычислительных сред, включая:

- архитектуру фон Неймана и её эволюцию;
- параллельные и распределённые системы (мультипроцессорные, кластерные, облачные архитектуры);
- специализированные процессоры (GPU, TPU) и их применение в AI и Big Data.

3.1.2. Технологии аналитических хранилищ данных (АХД) и обработки больших данных:

- принципы работы OLAP и OLTP-систем;
- технологии Hadoop, Spark, колоночных СУБД (ClickHouse, Greenplum);
- методы оптимизации запросов и индексирования.

3.1.3. Основы искусственного интеллекта и нейронных сетей в контексте вычислительных сред:

- архитектуры CNN, RNN, Transformer и их применение;
- особенности обучения и инференса моделей в распределённых средах;
- аппаратные требования для работы AI-алгоритмов.

3.1.4. Принципы работы автономных систем и робототехники:

- сенсорные системы (LIDAR, камеры, радары);
- алгоритмы компьютерного зрения и навигации;
- взаимодействие с системами управления в реальном времени.

3.2. Освоить навыки

3.2.1. Проектировать и анализировать архитектуру вычислительных сред для:

- обработки больших данных (выбор технологий, масштабирование);
- работы AI-моделей (оптимизация ресурсов CPU/GPU/TPU);
- автономных систем (расчёт latency, надёжности).

3.2.2. Работать с современными инструментами и платформами:

- развёртывать хранилища данных (HDFS, Apache Kafka);
- настраивать распределённое обучение нейросетей (TensorFlow/PyTorch);
- разрабатывать компоненты для робототехнических систем (ROS, NVIDIA Jetson).

3.2.3. Оптимизировать производительность систем:

- анализировать bottlenecks в Big Data-конвейерах;
- ускорять инференс моделей (квантование, pruning);
- настраивать взаимодействие сенсоров и ПО в реальном времени.

3.3. Овладеть техническими навыками

3.3.1. Навыками работы с:

- фреймворками для распределённых вычислений (Apache Spark, Dask);
- облачными платформами (для AI и IoT);
- симуляторами робототехники (Gazebo, CARLA).

3.3.2. Методами оценки эффективности вычислительных сред:

- нагрузочное тестирование (Big Data-кластеров, AI-инференса);
- анализ энергопотребления и latency (для edge-устройств);
- интерпретация метрик качества AI-моделей.

3.4. Общепрофессиональные компетенции (ОПК):

ОПК-2: Способен разрабатывать и развивать математические методы моделирования объектов, процессов и систем в области профессиональной деятельности.

ОПК-3: Способен разрабатывать наукоемкое программное обеспечение для автоматизации систем и процессов, а также развивать информационно-коммуникационные технологии.

3.5. Применение знаний

Результаты освоения дисциплины позволяют:

- разрабатывать компоненты для AI-стартапов (от дата-пайплайнов до embedded-решений);
- работать в областях: Data Engineering, ML Engineering, Robotics;
- продолжать исследования в магистратуре/аспирантуре (квантовые вычисления, нейроморфные системы).

4. Содержание дисциплины

4.1. Базовые концепции и архитектуры вычислительных сред

Базовые концепции и архитектуры вычислительных средств формируют теоретический фундамент современных компьютерных систем. Ключевыми принципами являются булева алгебра, лежащая в основе цифровой логики, теория автоматов, описывающая формальные модели вычислений, и архитектурные подходы к организации обработки информации. Среди разнообразия архитектур особое место занимает архитектура фон Неймана, которая уже более 75 лет остается доминирующей парадигмой вычислительных систем (рис.1).

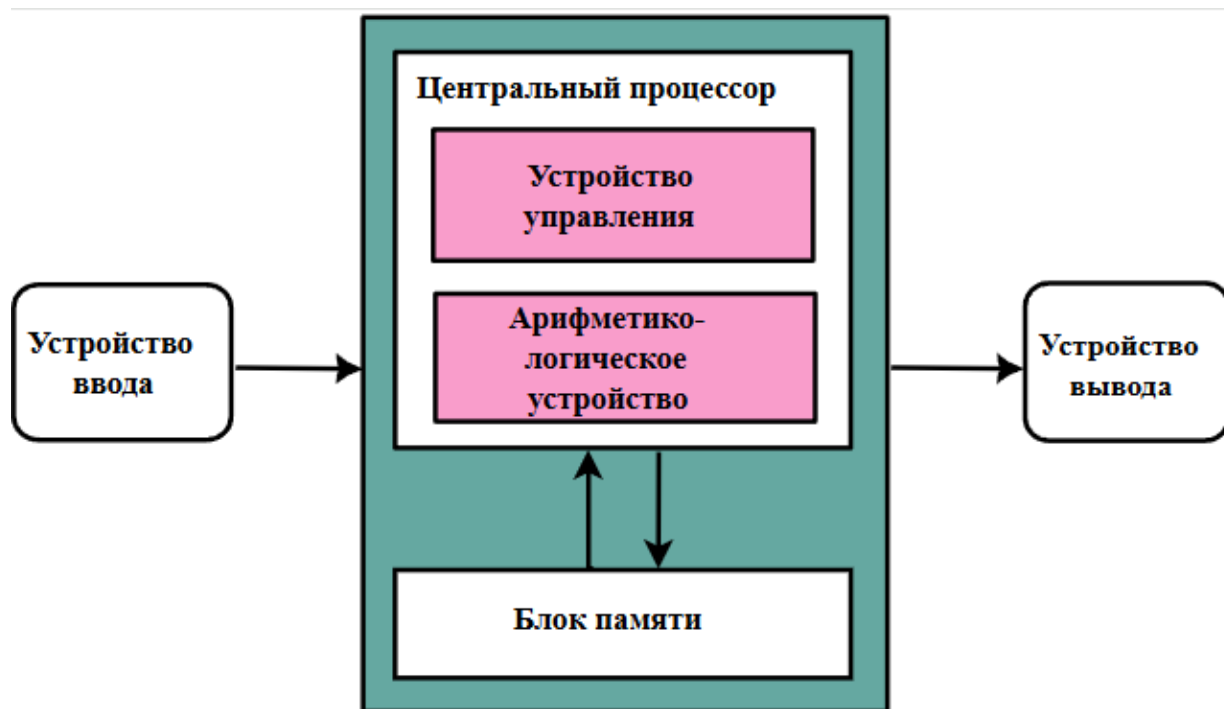


Рисунок 1 – Архитектура фон Неймана.

Ее универсальность проявляется в объединении процессора (CPU), единой памяти для хранения данных и команд, устройств ввода-вывода и шин передачи данных в единую систему. Характерной особенностью этой архитектуры является принцип хранимой программы, обеспечивающий

гибкость вычислений, но создающий известное "фон-неймановское бутылочное горлышко" при обмене данными между процессором и памятью.

Современные вычислительные средства демонстрируют эволюцию этих базовых концепций, что особенно наглядно видно при сравнении CPU и GPU. Если традиционные CPU (центральные процессоры) с их небольшим количеством мощных ядер (обычно до 64) оптимизированы для последовательного выполнения сложных инструкций, то GPU (графические процессоры), содержащие тысячи упрощенных ядер, идеально подходят для параллельной обработки однотипных операций (рис.2).

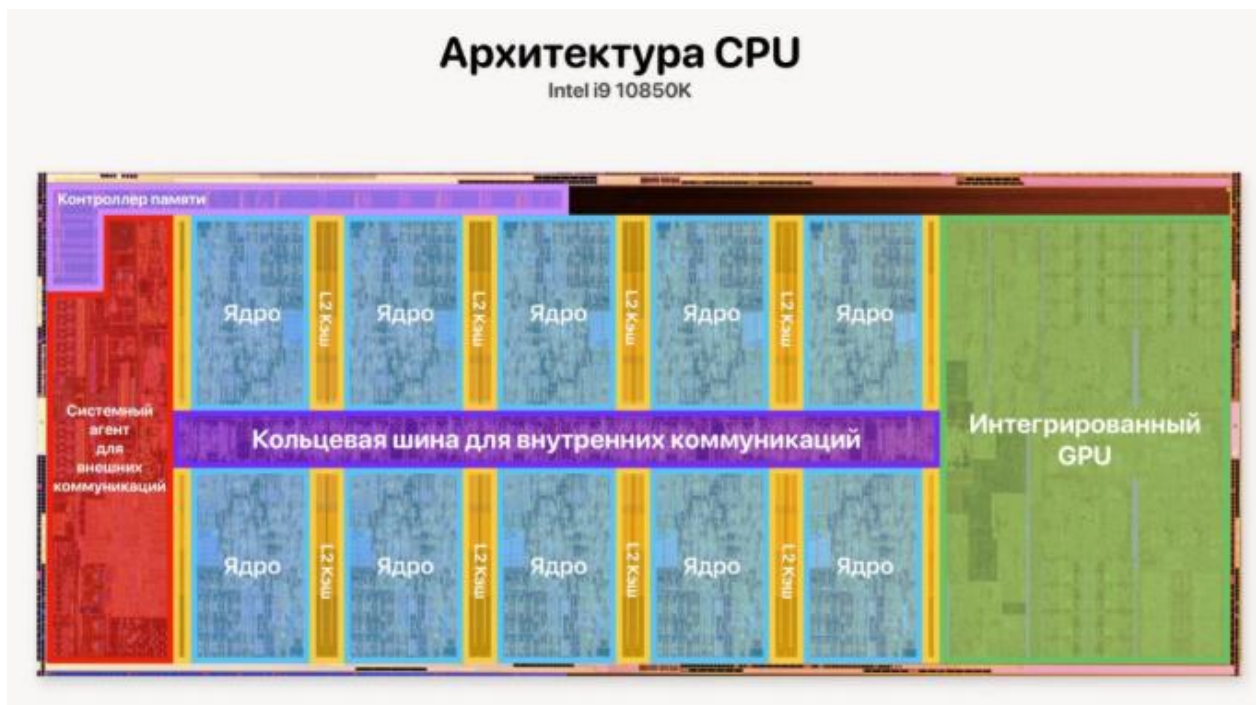


Рисунок 2 – Архитектура современного CPU процессора.

CPU содержит следующие компоненты (рис.2):

- несколько сложных ядер (от 1 до 24 в обычных домашних компьютерах);
- контроллер памяти для коммуникации с RAM;
- кэш трёх уровней (L1 в каждом ядре, L2 на несколько ядер, L3 на весь процессор);
- шина связи между компонентами (кольцевая шина);

- дополнительный блок для коммуникации чипа с остальными элементами компьютера (системный агент).

GPU имеет такие компоненты (рис.3):

- тысячи простых ядер;
- контроллер памяти для коммуникации с VRAM;
- кэш обычно двух уровней (L1 на ядро и L2 на процессор);
- планировщик для управления компонентами (поточковый мультипроцессор);
- интерфейс для коммуникации с CPU.

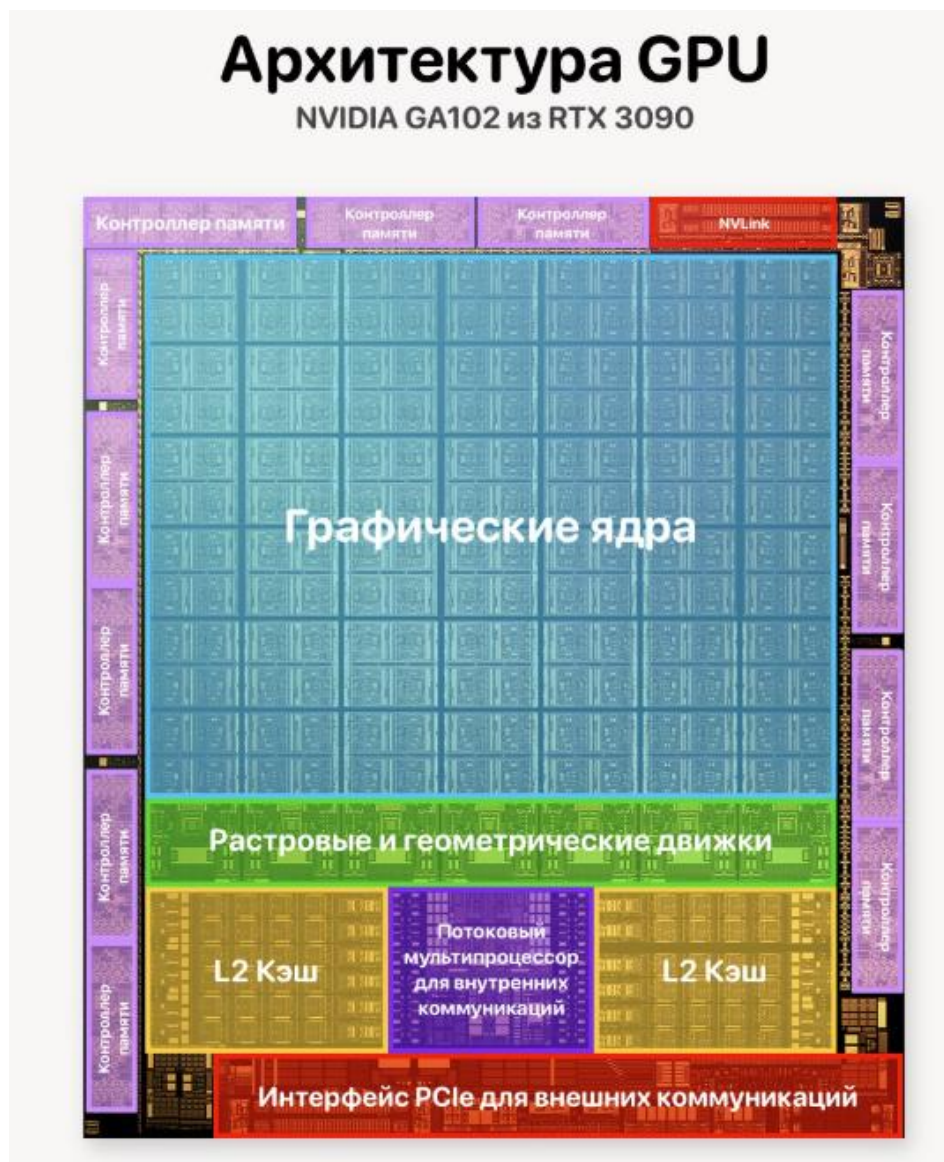


Рисунок 3 – Архитектура современного GPU процессора.

Это различие в архитектурных подходах отражает ответ вычислительной техники на современные вызовы: CPU остаются "узкоспециализированными мастерами" для сложных последовательных задач, тогда как GPU превратились в "армии рабочих" для массовой параллельной обработки данных. Особую значимость такая архитектурная специализация приобрела в эпоху big data и искусственного интеллекта, где GPU стали незаменимыми для выполнения матричных операций, лежащих в основе современных алгоритмов машинного обучения.

Высоконагруженные системы требуют принципиально иного архитектурного подхода по сравнению с традиционными вычислительными средами. Их проектирование строится на трех ключевых принципах: горизонтальном масштабировании (замена одного мощного сервера кластером из сотен узлов), оптимизации доступа к данным (использование кэшей Redis и колоночных СУБД вместо реляционных баз) и асинхронной обработке запросов через очереди сообщений типа Kafka. Эти решения позволяют обойти фундаментальные ограничения архитектуры фон Неймана, особенно проблему "бутылочного горлышка" при обмене данными между процессором и памятью. Современные подходы также активно используют гетерогенные вычисления, где разные задачи распределяются между CPU, GPU и специализированными ускорителями вроде TPU.

Геораспределенность стала обязательным требованием для truly высоконагруженных систем. Технологии edge-вычислений и CDN (например, Cloudflare) минимизируют задержки за счет размещения данных и вычислительных мощностей физически ближе к пользователям. Особое значение приобретают event-driven архитектуры и потоковая обработка данных, что особенно важно для финансовых систем, стриминговых платформ и IoT-решений. Все эти компоненты работают как единый механизм, где каждый элемент - от балансировщика нагрузки до специализированных процессоров - оптимизирован под конкретный тип workload.

4.2. Аналитические хранилища данных и Big Data

Аналитические хранилища данных (АХД) и технологии Big Data представляют собой критически важный компонент современного образования в области компьютерных наук и анализа данных. Для студентов изучение этих технологий — это не просто освоение конкретных инструментов вроде Hadoop или Spark, а формирование системного понимания методологии работы с большими данными. Методологический подход учит будущих специалистов правильно проектировать архитектуру хранения, выбирать оптимальные стратегии обработки информации и оценивать компромиссы между различными технологическими решениями. Эти навыки становятся фундаментом для работы в таких востребованных областях, как data engineering, business intelligence и системная аналитика.

Практическое значение изучения АХД проявляется в трех ключевых аспектах профессиональной подготовки. Во-первых, студенты учатся различать оперативную (OLTP) и аналитическую (OLAP) обработку данных, что является базисом для проектирования любых информационных систем. Во-вторых, работа с реальными кейсами из финансов, логистики и IoT формирует понимание отраслевой специфики — например, как ClickHouse ускоряет анализ фрод-транзакций в банках или как Data Lakes помогают обрабатывать телеметрию в умных городах. В-третьих, освоение методологии оптимизации запросов и индексации развивает критически важное "инженерное" мышление для решения проблем производительности. В таблице 1 приведены критерии сравнения OLTP и OLAP.

Таблица 1 – Сравнение OLTP и OLAP

Критерий	OLTP (Online Transaction Processing)	OLAP (Online Analytical Processing)
Основное назначение	Обработка операционных транзакций в реальном времени	Анализ больших объемов исторических данных

Тип запросов	Короткие, простые (вставка, обновление, удаление)	Сложные аналитические запросы с агрегацией
Частота операций	Очень высокая (тысячи транзакций в секунду)	Низкая (несколько сложных запросов в минуту/час)
Структура данных	Строчное хранение (row- oriented)	Колоночное хранение (column-oriented)
Оптимизация	Для скорости записи и коротких чтений	Для скорости выполнения аналитических запросов
Примеры СУБД	PostgreSQL, MySQL, Oracle	ClickHouse, Vertica, Greenplum
Размер данных	Относительно небольшие (гигабайты)	Очень большие (терабайты и петабайты)

Методологическая база курса специально структурирована от фундаментальных концепций к современным технологическим трендам. Сначала студенты осваивают базовые принципы колоночного хранения и параллельной обработки, затем переходят к изучению конкретных реализаций (Hadoop, Spark), и наконец — к комплексным case studies. Такой подход позволяет не просто механически изучить технологии, но понять их место в экосистеме Big Data и научиться адаптировать архитектурные решения под конкретные бизнес-задачи. Особый акцент делается на сравнении различных подходов, чтобы развить у студентов способность к взвешенному технологическому выбору.

4.3. Искусственный интеллект и нейронные сети в вычислительных средах

Современные вычислительные среды кардинально трансформируются под влиянием технологий искусственного интеллекта и нейронных сетей, что требует принципиально новых методологических подходов к их проектированию и эксплуатации. В отличие от традиционных

алгоритмических парадигм, нейросетевые модели демонстрируют принципиально иную природу вычислительных процессов - массово параллельную обработку матричных операций с крайне высокой степенью недетерминированности. Это обуславливает необходимость переосмысления классических архитектурных принципов фон Неймана, поскольку современные ИИ-системы требуют специализированных вычислительных сред с особыми характеристиками памяти, топологиями межпроцессорного взаимодействия и механизмами управления потоками данных.

Методологический анализ нейросетевых вычислений выявляет три ключевых аспекта их интеграции в вычислительные среды: аппаратную специализацию, распределённую обработку и программно-аппаратную ко-оптимизацию. Аппаратная специализация проявляется в появлении новых классов процессоров (GPU, TPU, нейроморфные чипы), оптимизированных именно для матричных операций, составляющих основу нейросетевых вычислений. Распределённая обработка становится обязательным требованием для обучения крупных моделей, что порождает новые методологические вызовы в области балансировки нагрузки, синхронизации параметров и отказоустойчивости. Ко-оптимизация программных алгоритмов и аппаратных архитектур приводит к возникновению новых парадигм, таких как "вычисления в памяти" (in-memory computing) и "приближённые вычисления" (approximate computing).

С методологической точки зрения, проектирование вычислительных сред для ИИ требует комплексного учёта всего технологического стека - от физического уровня (оптимизация микросхем для матричных операций) до системного (оркестрация распределённых вычислений) и прикладного (интеграция с бизнес-процессами). Особое значение приобретают вопросы энергоэффективности, поскольку обучение современных нейросетевых моделей сопряжено с огромными вычислительными затратами. Методология MLOps (Machine Learning Operations) становится связующим звеном между разработкой моделей и их промышленной эксплуатацией, предлагая

стандартизированные подходы к развёртыванию, мониторингу и обновлению ИИ-систем в различных вычислительных средах - от облачных платформ до периферийных устройств (рис.4).



Рисунок 4 – Основные элементы MLOps.

В рамках курса применяется трехуровневый методологический подход: от фундаментальных концепций к прикладным решениям. На первом уровне студенты осваивают математические основы (градиентный спуск, backpropagation) через практические лабораторные работы с линейными моделями. Второй уровень предполагает сравнительный анализ алгоритмов, где методология "от простого к сложному" позволяет понять эволюцию от линейной регрессии до многослойных перцептронов. Ключевой акцент делается на методологии оценки моделей - разделение выборок, кросс-валидация и интерпретация метрик качества, что формирует строгий научный подход к построению ML-решений. Третий уровень интегрирует полученные знания в сквозные кейсы, демонстрируя полный цикл CRISP-DM (Cross-Industry Standard Process for Data Mining).

Курс использует систематизированную методологию анализа нейросетевых архитектур через призму "проблема-решение". Для каждого

типа данных (изображения, тексты, временные ряды) строится сравнительная матрица:

- 1) Характеристики входных данных.
- 2) Инвариантности (трансляционная для CNN, временная для RNN).
- 3) Соответствующие архитектурные паттерны.

Методология "архитектурного прототипирования" позволяет студентам на практике оценивать компромиссы между сложностью модели и её производительностью. Особое внимание уделяется методологии переноса обучения (transfer learning), где отрабатываются навыки адаптации предобученных моделей к конкретным прогнозным задачам в финансах, метеорологии и социальной аналитике.

Курс реализует практико-ориентированную методологию освоения распределённых вычислений через три последовательных этапа:

- 1) Локальная оптимизация - профилирование ресурсов (CPU/GPU/TPU utilization).
- 2) Горизонтальное масштабирование - освоение техник data parallelism (PyTorch DDP).
- 3) Гибридные стратегии - комбинация model и pipeline parallelism.

Методология основана на принципе "fail-fast" - студенты учатся диагностировать узкие места (communication overhead, stragglers) через серию контролируемых экспериментов. Для инференса применяется методология A/B-тестирования моделей в продакшн-подобных условиях с использованием Kubernetes и специализированных serving-фреймворков. Сквозной методологический принцип - документирование trade-off между точностью, задержками и стоимостью вычислений.

4.4. Автономные системы и робототехника

В курсе применяется системная методология изучения автопилотов, основанная на декомпозиции сложных систем на функциональные модули: восприятие среды, принятие решений и исполнительные механизмы. Методология "digital twin" позволяет студентам сначала отрабатывать

алгоритмы в симулируемой среде (автоВАЗ), а затем переносить решения на реальные платформы. Особый акцент делается на методологии верификации и валидации (V&V) автономных систем, включая анализ edge-cases и стресс-тестирование в контролируемых условиях. Практические задания строятся по принципу постепенного усложнения - от следования по линии до полноценного автономного навигационного задания (рис.5).

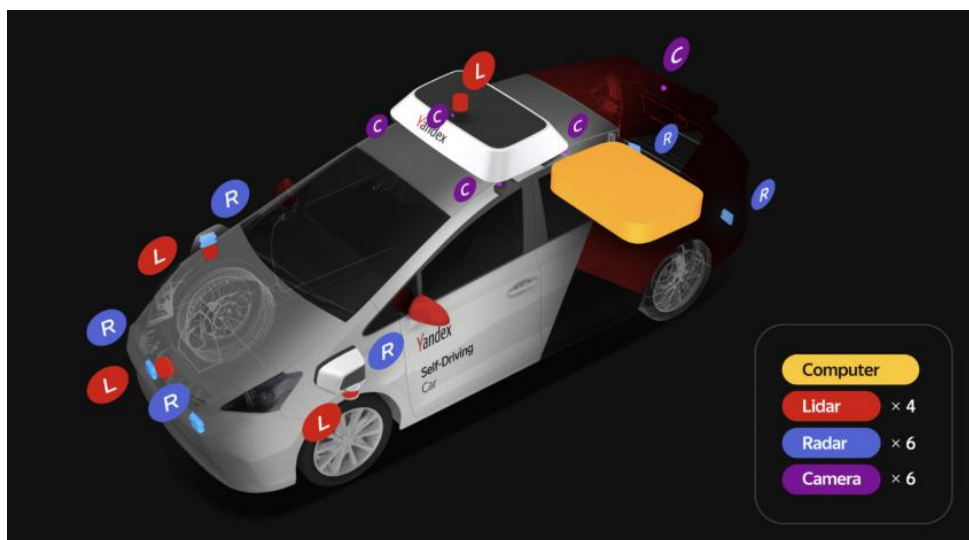


Рисунок 5 – Пример расположения устройств автопилотируемого транспорта.

Курс реализует комплексную методологию работы с сенсорными данными, основанную на pipeline'e "сырые данные → калибровка → фьюжн → интерпретация". Студенты осваивают методологию временной синхронизации (time alignment) данных от разнородных источников (LIDAR, камер, радаров) с использованием ROS-топиков. Методология "sensor fusion" преподаётся через сравнительный анализ подходов (Kalman filters, particle filters, deep learning-based), с обязательной оценкой точности и отказоустойчивости каждого метода. Лабораторные работы включают создание цифровых фильтров для обработки зашумленных данных и методологию оценки качества сенсорных систем через метрики precision-recall.

В курсе применяется human-centered методология разработки интерфейсов, включающая:

- 1) Анализ требований и сценариев использования.
- 2) Прототипирование (Figma, RViz).
- 3) Юзабилити-тестирование.

Для реальных интерфейсов изучается методология fail-safe design и redundancy, для виртуальных - методология создания интуитивных digital twins. Особое внимание уделяется методологии оценки когнитивной нагрузки оператора и эргономическим принципам. Практикум включает A/B-тестирование различных интерфейсных решений с последующим анализом метрик эффективности управления.

4.5. Современные вычислительные среды для ИИ и робототехники

Современные технологии искусственного интеллекта и робототехники требуют гибких и масштабируемых вычислительных сред, способных адаптироваться к различным workload. Ключевым трендом стало разделение на облачные и edge-решения: облачные платформы (AWS, Google Cloud) обеспечивают неограниченные ресурсы для обучения сложных моделей, в то время как edge-устройства (NVIDIA Jetson, Raspberry Pi) позволяют выполнять инференс непосредственно на месте сбора данных. Такой гибридный подход особенно востребован в приложениях реального времени, где критически важны скорость отклика и автономность работы. Современные фреймворки (ROS 2, Isaac SDK) предоставляют инструменты для эффективного распределения задач между облаком и edge-устройствами.

Облачные платформы предлагают готовые сервисы для машинного обучения (AWS SageMaker, Google AI Platform), значительно ускоряющие разработку и развертывание моделей. В то же время edge-вычисления становятся все более популярными благодаря специализированным аппаратным решениям, таким как NVIDIA Jetson с поддержкой CUDA для ускорения нейросетевых вычислений. Методология "обучение в облаке - выполнение на edge" позволяет оптимизировать затраты и обеспечить работу

в условиях нестабильного интернет-соединения. Особое внимание уделяется вопросам энергоэффективности и теплового проектирования edge-устройств, что критически важно для автономных систем.

Реализация систем с минимальной задержкой требует комплексного подхода, сочетающего аппаратные и программные решения. Технологии 5G обеспечивают необходимую пропускную способность и низкий пинг для распределенных AI-приложений. Квантовые вычисления, хотя и находятся на ранних этапах развития, обещают революцию в решении оптимизационных задач, актуальных для робототехники и логистики. Особый интерес представляют нейроморфные процессоры, имитирующие работу биологических нейронных сетей и способные обрабатывать сенсорные данные с беспрецедентной эффективностью. Эти технологии открывают новые возможности для создания действительно автономных систем реального времени.

С развитием автономных систем все острее встают вопросы кибербезопасности и этического использования ИИ. В курс включены методологии защиты данных (homomorphic encryption, federated learning) и обеспечения устойчивости моделей к adversarial-атакам (рис.6).

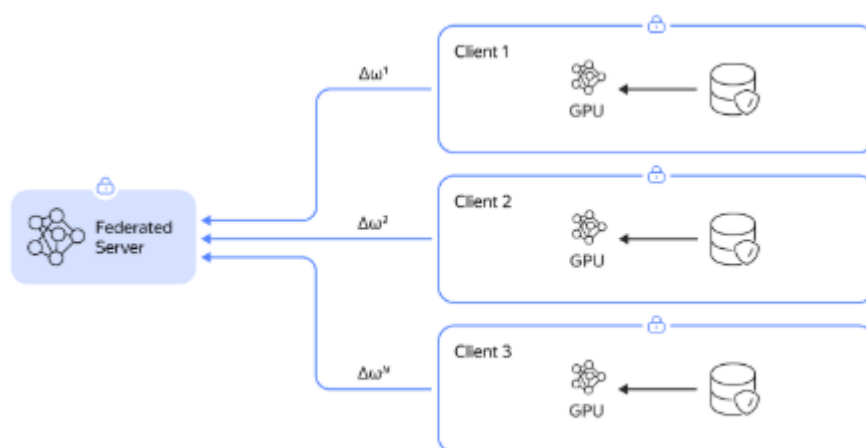


Рисунок 6 – Пример архитектуры федеративного обучения.

Особое внимание уделяется этическим аспектам - проблеме bias в данных, прозрачности принятия решений и ответственности за действия автономных систем. Студенты учатся применять frameworks для этического AI (например, Microsoft Responsible AI Principles) на всех этапах разработки - от сбора данных до развертывания моделей в production.

5. Образовательные технологии

Курс сочетает традиционные и инновационные форматы обучения, обеспечивая глубокое погружение в современные технологии. Лекции охватывают все ключевые аспекты — от основ архитектуры вычислительных систем до передовых решений в области AI и робототехники. Теоретический материал подкрепляется реальными кейсами из индустрии, что позволяет студентам сразу видеть практическое применение изучаемых концепций.

Активная практическая составляющая включает лабораторные работы, где студенты step-by-step проектируют и оптимизируют архитектуры, используя актуальные инструменты и фреймворки. Особое внимание уделяется командной работе: участие в хакатонах и кейс-стади, посвященных AI и робототехнике, не только развивает навыки, но и дает возможность проявить себя. За успешное выступление в таких мероприятиях, а также за публикацию научных статей по тематике курса, студенты могут получить автомат по дисциплине.

Один из ключевых элементов курса — индивидуальный проект, который каждый студент выполняет самостоятельно под руководством преподавателя. Проект предполагает разработку и реализацию решения, основанного на современных технологиях, будь то облачные вычисления, edge-устройства или распределенные системы. Это позволяет не только закрепить полученные знания, но и создать портфолио для будущей карьеры. Таким образом, курс построен так, чтобы совмещать теорию с практикой, командную работу с индивидуальными достижениями, а академические знания — с реальными вызовами индустрии. Такой подход гарантирует, что студенты не только усвоят материал, но и научатся применять его в профессиональной деятельности.

6. Методология оценки студента

Методологически система оценки построена на принципах комплексного и дифференцированного подхода, сочетающего академические достижения, практические навыки и инновационную деятельность. Педагогическая основа — "оценка для обучения" (assessment for learning), где баллы отражают не только результат, но и прогресс студента на всех этапах: от усвоения теории на лекциях до реализации проектов с элементами исследовательской работы. Ключевой акцент делается на практико-ориентированности — лабораторные и индивидуальный проект оцениваются строже, чем пассивное участие, чтобы мотивировать к применению знаний. Гибкость системы проявляется в бонусах за хакатоны и статьи, что поощряет инициативу и выход за рамки программы.

Ранжирование проектов (табл. 2) следует принципу "от простого к сложному", позволяя студентам выбирать задачи по уровню подготовки: например, классификация погоды — стартовый вариант, а RL для игрового ИИ — продвинутый. Критерии (параллелизм, юзерфрендли и др.) задают стандартизированные ориентиры качества, но не ограничивают творчество. Такой подход сочетает структурированность (четкие критерии) с индивидуализацией (выбор темы под интересы), что соответствует современным трендам персонализированного образования.

В таблице 3 представлены варианты систем для проектов.

Таблица 2 – Система оценки студента (максимум 100 баллов)

Критерий	Макс. балл	Описание
Активность на лекциях	15	Участие в обсуждениях, ответы на вопросы, работа в группах.
Лабораторные работы	25	Полнота выполнения, качество кода, защита работы.
Индивидуальный проект	30	Сложность, актуальность, реализация,

		документация.
Хакатоны/кейс-стади	10	Участие (+5), призовые места (+5).
Научные статьи/доклады	10	Публикация в сборнике конференции или выступление с докладом.
Доп. фиши в проекте	10	Внедрение инновационных решений (например, ML-оптимизация, асинхронность).
Автомат (бонус)	+	За 2 хакатона/статью или выдающийся проект.

Таблица 3 – Варианты систем для проектов

Тип нейронной сети	Система / Исходные данные	Технологии	Критерии системы
Классификация	Прогноз погоды по спутниковым данным	TensorFlow, RabbitMQ, Docker	Параллелизм, точность прогноза
Сегментация	Медицинские снимки (рентген/MPT)	PyTorch, FastAPI, Kubernetes	Быстрота обработки, дашборды
Регрессия	Прогноз цен на криптовалюты	Scikit-learn, Kafka, Grafana	Low-latency, ML-мониторинг
Детектирование объектов	Определение машины с городской камеры	YOLO, ROS, OpenCV	Real-time обработка, безопасность
Кластеризация	Анализ настроений в соцсетях	BERT, RabbitMQ, Elasticsearch	Масштабируемость, юзерфрендли
GAN (генерация)	Синтез изображений по заданной тематике	TensorFlow, Flask, Docker	Качество генерации, API-доступ
RNN (временные ряды)	Парсинг и анализ чатов (Telegram)	PyTorch, MongoDB, WebSockets	Анонимность, скорость инференса
Transformer (NLP)	Чат-бот для техподдержки	GPT-3, Django, Redis	Интеграция с CRM, ML-доработки
Reinforcement Learning	Игровой ИИ (стратегии/головоломки)	Unity ML-Agents, Python	Адаптивность, баланс сложности
Ансамбли моделей	Кредитный скоринг в банках	XGBoost, FastAPI, Prometheus	Интерпретируемость, надежность

7. Учебно-методическое обеспечение дисциплины

Курс обеспечен комплексом учебных и методических материалов, направленных на эффективное освоение теоретических и практических аспектов логики и архитектуры вычислительных сред. Основу составляют авторские лекционные презентации, включающие схемы, графики и примеры из реальных кейсов, а также лабораторные практикумы с пошаговыми инструкциями и заданиями разного уровня сложности. Для углубленного изучения тем предусмотрены электронные учебные пособия, содержащие теоретические выкладки, примеры кода и контрольные вопросы для самопроверки.

Важным элементом являются видеозаписи лекций и мастер-классов от экспертов индустрии, доступные в LMS (Moodle, OpenEdu), что позволяет организовать гибкое обучение, в том числе для студентов с особыми потребностями. Практическая часть подкреплена облачными sandbox-средами (Google Colab) и набором датасетов для тренировки моделей. Для проектной работы предоставляются шаблоны технической документации и чек-листы по код-ревью.

Дополнительно курс включает:

- глоссарий терминов с объяснением ключевых концепций;
- подборку научных статей (IEEE, Springer) по темам AI и распределенных систем;
- интерактивные симуляторы (например, автоВАЗ симулятор для автопилота).

8. Материально-техническая база

Данная дисциплина реализуется без предоставления специализированного оборудования со стороны учебного заведения. Все практические и проектные работы выполняются студентами на личных компьютерах с использованием общедоступных технологий и облачных сервисов. Такой подход развивает навыки самостоятельной организации рабочего процесса и адаптации к реальным условиям, где ресурсы часто ограничены.

Используемые студентами решения:

- Персональные устройства:
 - Ноутбуки/ПК с минимальными требованиями (поддержка Python, Docker);
 - Смартфоны для тестирования мобильных решений (при необходимости).
- Облачные и бесплатные сервисы:
 - Google Colab – для работы с нейросетями без GPU на локальной машине;
 - Github Codespaces – облачные среды разработки;
 - ROS на виртуальных машинах – для моделирования робототехники.
- Открытое ПО и датасеты:
 - Библиотеки: TensorFlow/PyTorch, OpenCV, Scikit-learn;
 - Публичные датасеты (Kaggle, UCI);
 - Симуляторы: автоВАЗ (автопилоты).
- Альтернативы специализированному оборудованию:
 - Вместо LIDAR – обработка данных с камер + алгоритмы SLAM;
 - Вместо промышленных датчиков – эмуляция данных или использование смартфонных сенсоров (акселерометр, GPS);
 - Вместо NVIDIA Jetson – оптимизация моделей для CPU (квантование, pruning).

9. Методические рекомендации

Ниже представлена таблица 4 с методическими рекомендациями и полезными ресурсами.

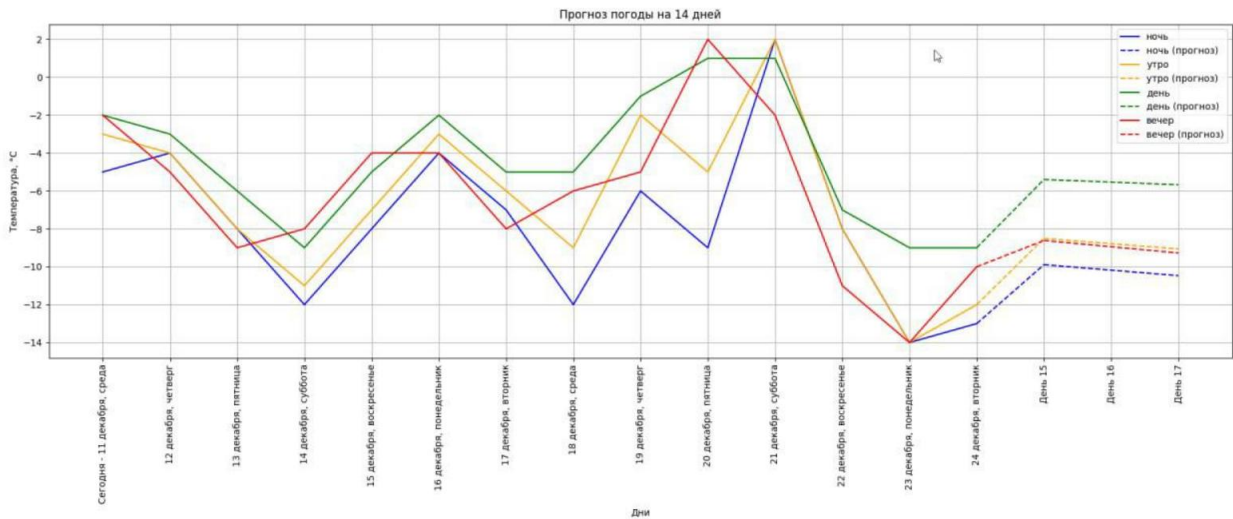
Таблица 4 - Методические рекомендации и полезные ресурсы

Категория	Ресурсы	Описание
Онлайн-курсы	Google Machine Learning Crash Course	Бесплатный курс по основам ML от Google.
	Stepik: "Введение в Data Science"	Интерактивный курс с задачами по Python и анализу данных.
	Coursera: "Deep Learning Specialization" (Andrew Ng)	Платная специализация, но можно получить фин. помощь.
Программирование	Kaggle Learn	Мини-курсы по Python, ML и обработке данных.
	LeetCode	Задачи по алгоритмам и структурам данных.
Книги	"Глубокое обучение" (Ian Goodfellow)	Классика по нейросетям (есть бесплатная онлайн-версия).
	"Архитектура компьютера" (Таненбаум)	Основы архитектуры ЭВМ.
Практика	Google Colab	Бесплатные GPU для обучения моделей.
	Hugging Face	Готовые модели NLP и датасеты.
	ROS Wiki	Документация по Robot Operating System.
Сообщества	Stack Overflow	Вопросы и ответы по программированию.
	Хабр	Статьи и обсуждения IT-тематики.
Доп. материалы	arXiv.org	Научные статьи по AI, ML, робототехнике.
	Towards Data Science	Блог с разборами ML-кейсов.

Приложения

Приложение 1

График предсказание погоды одного из студентов.



Список вопросов для зачета.

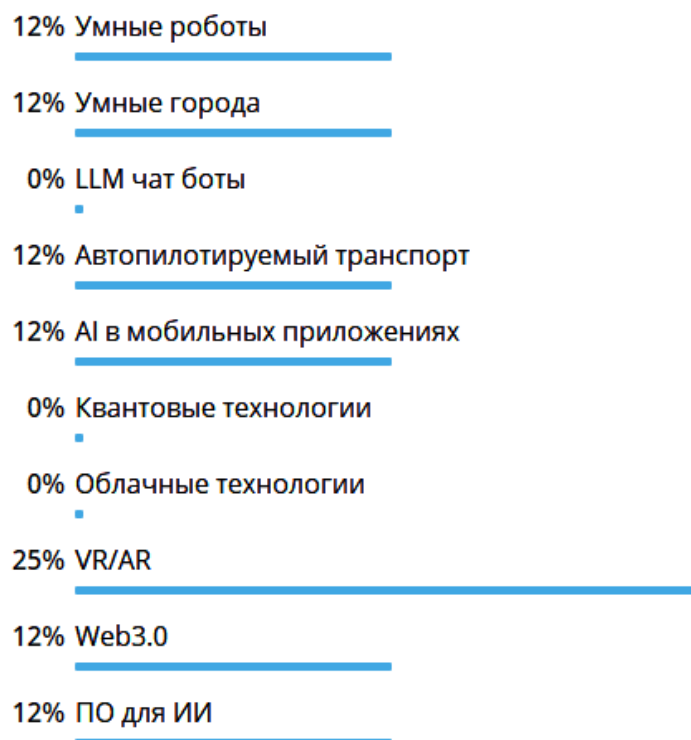
1. Что такое микросервисная архитектура и какие её основные принципы?
2. Какие технологии используются для оркестрации микросервисов (например, Kubernetes, Docker)?
3. Как обеспечить безопасность в микросервисной архитектуре?
4. Что такое высоконагруженные системы и какие их основные характеристики?
5. Какие технологии используются для балансировки нагрузки (например, NGINX, HAProxy)?
6. Как обеспечить масштабируемость высоконагруженных систем?
7. Какие методы кэширования используются в высоконагруженных системах (например, Redis, Memcached)?
8. Как обеспечить отказоустойчивость в высоконагруженных системах?
9. Какие паттерны проектирования используются в высоконагруженных системах?
10. Как реализовать асинхронную обработку в высоконагруженных системах?
11. Какие методы оптимизации производительности используются в высоконагруженных системах?
12. Какие основные компоненты архитектуры автопилота?
13. Какие технологии используются для обработки данных с сенсоров в автопилоте (например, LIDAR, RADAR)?
14. Как обеспечить безопасность и надежность автопилота?
15. Какие алгоритмы машинного обучения используются в автопилоте?
16. Как организовать взаимодействие между компонентами автопилота?
17. Какие методы обработки данных в реальном времени используются в автопилоте?
18. Как реализовать систему управления и контроля в автопилоте?

19. Что такое блокчейн и какие его основные принципы?
20. Какие типы блокчейнов существуют (например, публичные, частные, консорциумы)?
21. Какие технологии используются для создания смарт-контрактов (например, Ethereum, Hyperledger)?
22. Как обеспечить безопасность и конфиденциальность в блокчейн-системах?
23. Какие алгоритмы консенсуса используются в блокчейне (например, Proof of Work, Proof of Stake)?
24. Как реализовать систему управления доступом в блокчейн-системах?
25. Какие основные компоненты архитектуры VR/AR приложений?
26. Какие технологии используются для создания VR/AR приложений (например, Unity, Unreal Engine)?
27. Как обеспечить производительность и качество графики в VR/AR приложениях?
28. Какие методы взаимодействия с пользователем используются в VR/AR приложениях?
29. Как организовать взаимодействие между компонентами VR/AR приложений?
30. Как реализовать систему отслеживания движений в VR/AR приложениях?

Голосование за наиболее интересные проекты для студентов.

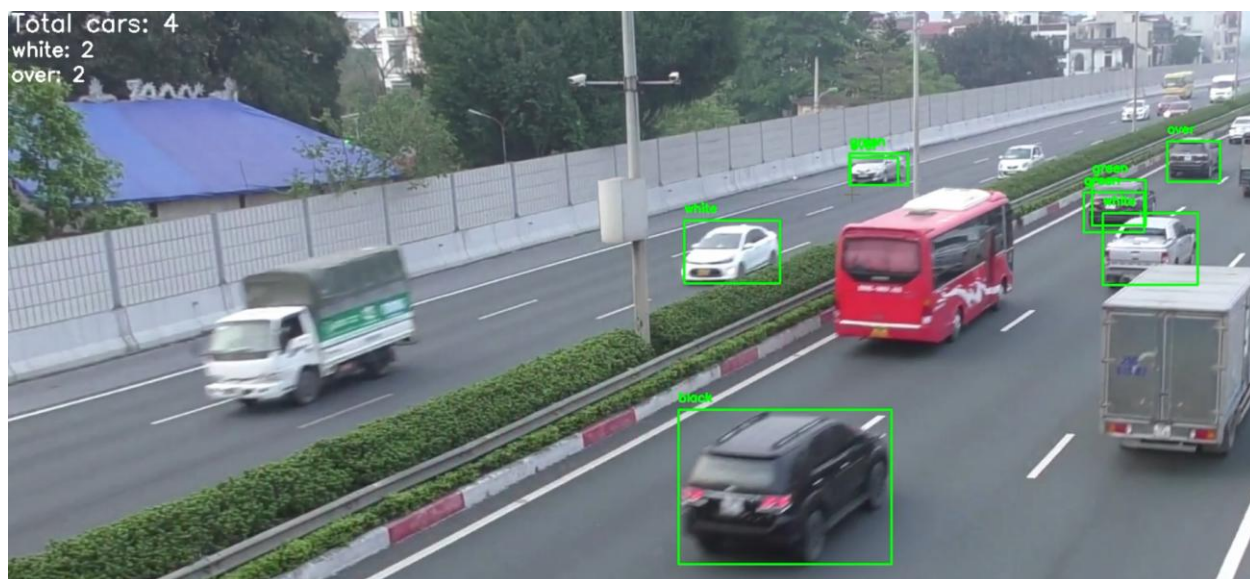
Голосуем за интересные проекты и их архитектуры

Анонимный опрос



Приложение 4

Проект подсчета машин определенных цветов одного из студентов.



Список литературы и полезных материалов

Основная литература:

1. Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвиль. – М.: ДМК Пресс, 2022. – 652 с. – ISBN 978-5-97060-554-4.
2. Оригинал: Goodfellow, I. Deep Learning / I. Goodfellow, Y. Bengio, A. Courville. – MIT Press, 2016.
3. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – 6-е изд. – СПб.: Питер, 2021. – 816 с. – ISBN 978-5-4461-1387-3.
4. Оригинал: Tanenbaum, A.S. Structured Computer Organization / A.S. Tanenbaum, T. Austin. – Pearson, 2012.
5. Рассел, С. Искусственный интеллект: современный подход / С. Рассел, П. Норвиг. – 4-е изд. – М.: Вильямс, 2021. – 1408 с. – ISBN 978-5-8459-1967-3.
6. Оригинал: Russell, S. Artificial Intelligence: A Modern Approach / S. Russell, P. Norvig. – Pearson, 2020.

Дополнительная литература:

1. Орехов, А.В. Распределенные системы. Принципы и парадигмы / А.В. Орехов. – М.: ДМК Пресс, 2020. – 512 с. – ISBN 978-5-93700-053-9.
2. Чарнецки, К. Шаблоны проектирования распределенных приложений / К. Чарнецки. – М.: Вильямс, 2019. – 368 с. – ISBN 978-5-8459-2076-1.