МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных технологий и прикладной математики

УТВЕРЖДАЮ:
Проректор по учебной работе, качеству образования – первый проректор
_____Хагуров Т.А.
подпись
« 29 » августа 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1. О.32 Параллельное и низкоуровневое программирование

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Профиль Искусственный интеллект и аналитика данных

Форма обучения очная

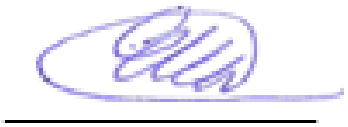
Квалификация бакалавр

Краснодар 2025

Рабочая программа дисциплины «Параллельное и низкоуровневое программирование» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем.

Программу составил(и):
Т.А. Приходько, доцент КВТ

Программу составил(и):
Е.А. Нигодин, ст. преп. КВТ
И.О. Фамилия, должность, ученая степень, ученое звание



подпись

Рабочая программа дисциплины «Параллельное и низкоуровневое программирование» утверждена на заседании кафедры вычислительных технологий протокол № 1 «26» _августа_2025 г.
И.О. заведующего кафедрой (разработчика) Т.А. Приходько

подпись

Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики
протокол № 1 «28» _августа_2025 г.
Председатель УМК факультета А. В. Коваленко

подпись

Рецензенты:

Мостовой Евгений Викторович, генеральный директор ООО «Портал-Юг», e-mail: mostovoy@portal-yug.ru

Луценко Евгений Вениаминович, доктор экономических наук, кандидат технических наук, профессор кафедры компьютерных технологий и систем Федерального государственного бюджетное образовательное учреждение высшего образования «Кубанский государственный аграрный университет имени И.Т. Трубилина», e-mail: prof.lutsenko@gmail.com

1 Цели и задачи изучения дисциплины (модуля)

1.1 Цель освоения дисциплины

Формирование у студентов знаний и практических навыков разработки многопоточных и высокопроизводительных приложений на языке C++ с использованием технологий POSIX Threads, OpenMP, CUDA, ROCm и OpenCL. Освоение архитектурных принципов параллельных вычислений на CPU и GPU, а также низкоуровневой оптимизации под современные процессоры (SSE, AVX).

1.2 Задачи дисциплины

1. Изучить основные принципы параллельного программирования и синхронизации потоков.
2. Освоить API POSIX Threads и средства параллелизма C++.
3. Научиться использовать OpenMP для распараллеливания вычислений на уровне потоков и циклов.
4. Изучить принципы векторизации вычислений с применением SSE и AVX-инструкций.
5. Освоить основы GPU-программирования на CUDA, ROCm и OpenCL.
6. Развить навыки профилирования и оптимизации кода под архитектуру CPU и GPU.

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Параллельное и низкоуровневое программирование» относится к «Часть, формируемая участниками образовательных отношений» Блока 1 «Дисциплины (модули)» учебного плана.

Для изучения дисциплины студент должен владеть знаниями, умениями и навыками по дисциплинам: Программирование, Алгоритмы и структуры данных, Операционные системы, с которыми дисциплина связана логически и содержательно-методически.

Дисциплина в значительной степени взаимодействует для формирования компетенций с дисциплинами:

- Технологии обработки больших данных;
- Современные методы компьютерного зрения;
- Подготовка данных машинного обучения;
- ИИ в робототехнике.

1.4 Профессиональные роли в структуре образовательной программы

Роль 1: Data Analyst (Аналитик данных)
Задачи:
1. Статистический анализ, визуализация данных, предварительная обработка.
2. Создание прогнозных моделей
3. Построение аналитических моделей для поддержки бизнес-решений.
Роль 2: MLOps (Специалист по эксплуатации ИИ)
Задачи:
1. DevOps для ML.
2. Автоматизация, мониторинг ML-систем.
3. Операционное управление жизненным циклом ML-моделей.
Роль 3: AI PM (Менеджер проектов ИИ)

Задачи:
1. Управление ИИ-проектами от идеи до внедрения
2. Анализ бизнес-требований и постановка задач
3. Оценка эффективности и ROI ИИ-решений

1.5 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Изучение данной учебной дисциплины направлено на формирование у обучающихся следующих компетенций:

ОПК-3	**Способен понимать и применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения**
ОПК-3.1	Аргументировано применяет современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения Умеет выбирать подходящие методы проектирования параллельных приложений, реализует лабораторные работы с pthreads, std::thread, OpenMP, CUDA Владеет способами решения и аргументированно объясняет по структуре программы и способам распараллеливания.
PL-1 Б	***Способен применять язык программирования Python для решения задач в области ИИ***
PL-1.4	***Проектирует системы распределённых вычислений на Python для эффективной обработки большого количества задач*** Оптимизирует код с использованием библиотек для научных вычислений. Способен применять основные функции фреймворка PySpark, может самостоятельно построить процесс обработки больших данных с использованием Airflow
PL-2 П	***(П) Способен применять JVM-совместимые языки программирования для решения задач в области ИИ***
PL-2.1	***Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений*** Понимает модель памяти Java и способен поддерживать приложения с высоким параллелизмом и конкуренцией. Понимает алгоритмы сборки мусора и способен оптимизировать сборку мусора.
PL-3 Б	***(Б) Способен применять языки программирования C/C++ для решения задач в области ИИ***
PL-3.1	***Разрабатывает и отлаживает эффективные многопоточные решения на C++, тестирует, испытывает и оценивает качество таких решений*** Знает основы синтаксиса языка. Знает общие принципы параллельных вычислений и понимает проблемы, возникающие при распараллеливании алгоритмов. Проводит распараллеивание простого алгоритма с применением OpenMP, стандартных библиотек C/C++ или др. Решает проблемы одновременного доступа к данным из нескольких потоков, грамотно применяет атомарные операции и механизм блокировок. Оценивает производительность, умеет профилировать код и устраняет найденные узкие места. Понимает возможности и ограничения встроенных систем. Находит и использует библиотеки, соответствующие решаемой задаче.

Знает основы синтаксиса языка C/C++, основы построения систем ИИ, общие принципы параллельных вычислений.
Умеет использовать средства отладки и профилирования кода, находить участки кода, ограничивающие производительность системы.

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 2 зач. ед. (72 часа), их распределение по видам работ представлено в таблице

Вид учебной работы		Всего часов	Семестры (часы)					
			4					
Контактная работа, в том числе:		**34,2**	**34,2**					
Аудиторные занятия (всего):		**32**	**32**					
Занятия лекционного типа		**16**	16					
Лабораторные занятия		**16**	16					
Занятия семинарского типа (семинары, практические занятия)								
Иная контактная работа:		**2,2**	**2,2**					
Контроль самостоятельной работы (КСР)		**2**	2					
Промежуточная аттестация (ИКР)		**0,2**	0,2					
Самостоятельная работа, в том числе:		**37,8**	**37,8**					
Курсовая работа								
Проработка учебного (теоретического) материала		**10**	10					
Выполнение индивидуальных заданий (подготовка сообщений, презентаций)		**27,8**	27,8					
Реферат								
Подготовка к текущему контролю								
Контроль:		**0**	**0**					
Подготовка к экзамену		**0**	**0**					
Общая трудоемкость	**час.**	**72**	**72**					
	в том числе контактная работа	**34,2**	**34,2**					
	зач. ед	**2**	**2**					

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.
Разделы (темы) дисциплины, изучаемые в 4 семестре

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
1.	Введение в параллельное программирование. Потоки и процессы в ОС.	8	2		2	4
2.	POSIX Threads (pthreads): создание, управление и синхронизация потоков.	8	2		2	4
3.	Параллелизм в C++: std::thread, mutex, future, async.	8	2		2	4

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
4.	OpenMP: директивы распараллеливания, секции, редукции, профилирование	8	2		2	4
5.	Векторизация и оптимизация вычислений: SIMD, SSE, AVX.	8	2		2	4
6.	GPU-программирование: основы архитектуры CUDA, модель потоков и памяти.	8	2		2	4
7.	OpenCL и ROCm: кроссплатформенные вычисления на GPU.	8	2		2	4
8.	Оптимизация и профилирование параллельных приложений на CPU и GPU.	13,8	2		2	9,8
ИТОГО по разделам дисциплины		**69,8**	**16**		**16**	**37,8**
Контроль самостоятельной работы (КСР)		2				
Промежуточная аттестация (ИКР)		0,2				
Подготовка к текущему контролю		0				
Общая трудоемкость по дисциплине		**72**				

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

2.3 Содержание разделов (тем) дисциплины

2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
1.	Введение в параллельное программирование. Потоки и процессы в ОС.	Модель процессов и потоков. Различие потоков и процессов. Преимущества и риски параллельного исполнения. Основы многопоточности и распределённых вычислений. Архитектуры CPU (SMP, NUMA). Введение в понятие *race condition* и *deadlock*.	ЛР
2.	POSIX Threads (pthreads): создание, управление и синхронизация потоков.	Структура POSIX Threads API. Создание потоков (pthread_create), завершение (pthread_join). Механизмы синхронизации (mutex, condition variable, barrier). Примеры использования в задачах суммирования, сортировки и поиска.	ЛР
3.	Параллелизм в C++: std::thread, mutex, future, async.	std::thread, std::mutex, std::lock_guard, std::unique_lock. Механизмы передачи данных между потоками: std::promise, std::future, std::async. Паттерны параллелизма: producer-consumer, map-reduce.	ЛР

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
4.	OpenMP: директивы распараллеливания, секции, редукции, профилирование	Принципы OpenMP. Основные директивы (parallel, for, sections, critical, reduction). Управление количеством потоков. Синхронизация и разделение переменных (shared, private). Практические примеры ускорения матричных операций.	ЛР
5.	Векторизация и оптимизация вычислений: SIMD, SSE, AVX.	Архитектура SIMD. Модель регистров SSE и AVX. Векторные типы данных и операции: сложение, умножение, редукция. Использование intrinsic-функций и авто-векторизации компилятора. Сравнение эффективности последовательного и векторного кода.	ЛР
6.	GPU-программирование: основы архитектуры CUDA, модель потоков и памяти.	Архитектура GPU: блоки, варпы, потоки. Модель памяти CUDA: global, shared, local, constant. Написание простейших ядер (global__ функции). Конфигурация сетки (<<<grid, block>>>). Различия между CPU и GPU-параллелизмом.	ЛР
7.	OpenCL и ROCm: кроссплатформенные вычисления на GPU.	Концепция OpenCL: платформы, устройства, контексты, очереди команд. Написание и компиляция ядра OpenCL. Управление памятью (clCreateBuffer, clEnqueueNDRangeKernel). ROCm — экосистема параллельных вычислений AMD. Сравнение производительности с CUDA.	ЛР
8.	Оптимизация и профилирование параллельных приложений на CPU и GPU.	Принципы оптимизации вычислительного кода: уменьшение накладных расходов синхронизации, улучшение кэш-локальности, балансировка нагрузки. Использование инструментов профилирования (perf, valgrind, nvprof, Nsight). Разбор реальных кейсов оптимизации.	ЛР

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.2 Занятия семинарского типа

Не предусмотрены

2.3.3 Лабораторные занятия

Лабораторные занятие состоят в выполнении индивидуальных заданий и объединены в 8 индивидуальных заданий

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
1.	Введение в параллельное программирование. Потоки и процессы в ОС.	Изучение API потоков. Создание программы, которая порождает несколько потоков для выполнения независимых задач (например, вычисление суммы и среднего массива). Измерение ускорения. Анализ количества потоков и влияния контекстных переключений.	ЛР
2.	POSIX Threads (pthreads): создание, управление и синхронизация потоков.	Реализация параллельного вычисления суммы элементов массива с использованием pthread_create, pthread_join. Добавление синхронизации через pthread_mutex_t и pthread_barrier_t. Измерение масштабируемости при увеличении числа потоков.	ЛР
3.	Параллелизм в C++: std::thread, mutex, future, async.	Реализация параллельной сортировки массива (merge sort или quick sort) с использованием std::thread и std::async. Измерение времени выполнения. Сравнение с POSIX Threads.	ЛР
4.	OpenMP: директивы распараллеливания, секции, редукции, профилирование	Использование директив #pragma omp parallel for для распараллеливания матричного умножения. Добавление reduction и schedule(dynamic). Сравнение производительности разных стратегий планирования.	ЛР
5.	Векторизация и оптимизация вычислений: SIMD, SSE, AVX.	Использование intrinsic-функций _mm256_add_ps, _mm256_mul_ps, _mm256_fmadd_ps для реализации векторного умножения и сложения массивов. Сравнение производительности с обычным циклом. Проверка влияния выравнивания памяти.	ЛР
6.	GPU-программирование: основы архитектуры CUDA, модель потоков и памяти.	Реализация CUDA-ядра для умножения векторов и матриц. Настройка конфигурации сетки (<<<grid, block>>>), работа с глобальной памятью, измерение ускорения относительно CPU.	ЛР
7.	OpenCL и ROCm: кроссплатформенные вычисления на GPU.	Реализация программы OpenCL для векторного сложения. Создание контекста, очереди команд, компиляция ядра. Сравнение времени выполнения с CUDA. Тестирование на CPU и GPU.	ЛР

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
8.	Оптимизация и профилирование параллельных приложений на CPU и GPU.	Использование perf, valgrind, nvprof, Nsight Systems для анализа «горячих» участков кода. Оптимизация вычислений: кэш-локальность, размер блоков, ограничение синхронизации. Отчёт о достигнутом ускорении и эффективности.	ЛР

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.4 Примерная тематика курсовых работ (проектов)

Не предусмотрены

2.4 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Вид СРС	Перечень учебно-методического обеспечения дисциплины по выполнению самостоятельной работы
1	2	3
1	Проработка и повторение лекционного материала, материала учебной и научной литературы, подготовка к семинарским занятиям	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой вычислительных технологий факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 07.05.2025 г.
2	Решение задач	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой вычислительных технологий факультета компьютерных технологий и прикладной математики ФГБОУ ВО «КубГУ», протокол №7 от 07.05.2025 г.

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ограниченными возможностями здоровья (ОВЗ) предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа,

– в форме аудиофайла,

– в печатной форме на языке Брайля.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа,

– в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

3. Образовательные технологии

В соответствии с требованиями ФГОС в программа дисциплины предусматривает использование в учебном процессе следующих образовательные технологии: чтение лекций с использованием мультимедийных технологий; метод малых групп, разбор практических задач и кейсов.

При обучении используются следующие образовательные технологии:

- Технология коммуникативного обучения – направлена на формирование коммуникативной компетентности студентов, которая является базовой, необходимой для адаптации к современным условиям межкультурной коммуникации.

- Технология разноуровневого (дифференцированного) обучения – предполагает осуществление познавательной деятельности студентов с учётом их индивидуальных способностей, возможностей и интересов, поощряя их реализовывать свой творческий потенциал. Создание и использование диагностических тестов является неотъемлемой частью данной технологии.

- Технология модульного обучения – предусматривает деление содержания дисциплины на достаточно автономные разделы (модули), интегрированные в общий курс.

- Информационно-коммуникационные технологии (ИКТ) - расширяют рамки образовательного процесса, повышая его практическую направленность, способствуют интенсификации самостоятельной работы учащихся и повышению познавательной активности. В рамках ИКТ выделяются 2 вида технологий:

- Технология использования компьютерных программ – позволяет эффективно дополнить процесс обучения языку на всех уровнях.

- Интернет-технологии – предоставляют широкие возможности для поиска информации, разработки научных проектов, ведения научных исследований.

- Технология индивидуализации обучения – помогает реализовывать личностно-ориентированный подход, учитывая индивидуальные особенности и потребности учащихся.

- Проектная технология – ориентирована на моделирование социального взаимодействия учащихся с целью решения задачи, которая определяется в рамках профессиональной подготовки, выделяя ту или иную предметную область.

- Технология обучения в сотрудничестве – реализует идею взаимного обучения, осуществляя как индивидуальную, так и коллективную ответственность за решение учебных задач.

- Игровая технология – позволяет развивать навыки рассмотрения ряда возможных способов решения проблем, активизируя мышление студентов и раскрывая личностный потенциал каждого учащегося.

- Технология развития критического мышления – способствует формированию разносторонней личности, способной критически относиться к информации, умению отбирать информацию для решения поставленной задачи.

Комплексное использование в учебном процессе всех вышеназванных технологий стимулируют личностную, интеллектуальную активность, развивают познавательные процессы, способствуют формированию компетенций, которыми должен обладать будущий специалист.

Основные виды интерактивных образовательных технологий включают в себя:

- работа в малых группах (команде) - совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путём творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности;

- проектная технология - индивидуальная или коллективная деятельность по отбору, распределению и систематизации материала по определенной теме, в результате которой составляется проект;

- анализ конкретных ситуаций - анализ реальных проблемных ситуаций, имевших место в соответствующей области профессиональной деятельности, и поиск вариантов лучших решений;

- развитие критического мышления – образовательная деятельность, направленная на развитие у студентов разумного, рефлексивного мышления, способного выдвинуть новые идеи и увидеть новые возможности.

Подход разбора конкретных задач и ситуаций широко используется как преподавателем, так и студентами во время лекций, лабораторных занятий и анализа результатов самостоятельной работы. Это обусловлено тем, что при исследовании и решении каждой конкретной задачи имеется, как правило, несколько методов, а это требует разбора и оценки целой совокупности конкретных ситуаций.

Темы, задания и вопросы для самостоятельной работы призваны сформировать навыки поиска информации, умения самостоятельно расширять и углублять знания, полученные в ходе лекционных и практических занятий.

Подход разбора конкретных ситуаций широко используется как преподавателем, так и студентами при проведении анализа результатов самостоятельной работы.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Для лиц с нарушениями зрения:
– в печатной форме увеличенным шрифтом,
– в форме электронного документа.
Для лиц с нарушениями слуха:
– в печатной форме,
– в форме электронного документа.
Для лиц с нарушениями опорно-двигательного аппарата:
– в печатной форме,
– в форме электронного документа.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4. Оценочные и методические материалы

4.1 Оценочные средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «название дисциплины».

Оценочные средства включает контрольные материалы для проведения **текущего контроля** в форме тестовых заданий, кейсов и **промежуточной аттестации** в форме вопросов и заданий к **экзамену**.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Структура оценочных средств для текущей и промежуточной аттестации

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
1	Введение в параллельное программирование. Потоки и процессы в ОС.	ОПК-3	*Лабораторная работа №1*	*Вопросы к зачету*
2	POSIX Threads (pthreads): создание, управление и синхронизация потоков.	ОПК-3, PL-1	*Лабораторная работа №2*	*Вопросы к зачету*
3	Параллелизм в C++: std::thread, mutex, future, async.	PL-3	*Лабораторная работа №3*	*Вопросы к зачету*
4	OpenMP: директивы распараллеливания, секции, редукции, профилирование	PL-2	*Лабораторная работа №4*	*Вопросы к зачету*
5	Векторизация и оптимизация вычислений: SIMD, SSE, AVX.	PL-2, ОПК-3	*Лабораторная работа №5*	*Вопросы к зачету*
6	GPU-программирование: основы архитектуры CUDA, модель потоков и памяти.	ОПК-3, PL-3	*Лабораторная работа №6*	*Вопросы к зачету*
7	OpenCL и ROCm: кроссплатформенные вычисления на GPU.	PL-3	*Лабораторная работа №7*	*Вопросы к зачету*

8	Оптимизация и профилирование параллельных приложений на CPU и GPU.	PL-3	*Лабораторная работа №8*	*Вопросы к зачету*

Показатели, критерии и шкала оценки сформированных компетенций

Соответствие **пороговому уровню** освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **удовлетворительно /зачтено**):

ОПК-3 **Способен понимать и применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения**

ОПК-3.1 Аргументировано применяет современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения
Умеет выбирать подходящие методы проектирования параллельных приложений, реализует лабораторные работы с pthreads, std::thread, OpenMP, CUDA
Владеет способами решения и аргументированно объясняет по структуре программы и способам распараллеливания.

PL-1 Б ***Способен применять язык программирования Python для решения задач в области ИИ***

PL-1.4 ***Проектирует системы распределённых вычислений на Python для эффективной обработки большого количества задач***
Оптимизирует код с использованием библиотек для научных вычислений.
Способен применять основные функции фреймворка PySpark, может самостоятельно построить процесс обработки больших данных с использованием Airflow

PL-2 П ***(П) Способен применять JVM-совместимые языки программирования для решения задач в области ИИ***

PL-2.1 ***Разрабатывает и отлаживает прикладные решения разного уровня сложности и для широкого круга конечных пользователей с использованием JVM-совместимых языков программирования, тестирует, испытывает и оценивает качество таких решений***
Понимает модель памяти Java и способен поддерживать приложения с высоким параллелизмом и конкуренцией. Понимает алгоритмы сборки мусора и способен оптимизировать сборку мусора.

PL-3 Б ***(Б) Способен применять языки программирования C/C++ для решения задач в области ИИ***

PL-3.1 ***Разрабатывает и отлаживает эффективные многопоточные решения на C++, тестирует, испытывает и оценивает качество таких решений***
Знает основы синтаксиса языка.
Знает общие принципы параллельных вычислений и понимает проблемы, возникающие при распараллеливании алгоритмов.
Проводит распараллеивание простого алгоритма с применением OpenMP, стандартных библиотек C/C++ или др.
Решает проблемы одновременного доступа к данным из нескольких потоков, грамотно применяет атомарные операции и механизм блокировок.
Оценивает производительность, умеет профилировать код и устраняет найденные узкие места.
Понимает возможности и ограничения встроенных систем.
Находит и использует библиотеки, соответствующие решаемой задаче.

Знает основы синтаксиса языка C/C++, основы построения систем ИИ, общие принципы параллельных вычислений.
Умеет использовать средства отладки и профилирования кода, находить участки кода, ограничивающие производительность системы.

Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Пример лабораторной работы

Лабораторная работа №1: Изучение API потоков в C/C++

«Многопоточные вычисления суммы и среднего массива»

Задача: создать программу, которая порождает несколько потоков для выполнения независимых задач (вычисление суммы и среднего массива). Измерить ускорение.

Что нужно сделать:

- Реализовать программу с использованием потоков;

- Измерить ускорение при различном количестве потоков;

- Проанализировать влияние контекстных переключений.

Инструменты: C/C++, POSIX Threads (pthread), стандартная библиотека C++, таймеры, MS Visual Studio / GCC.

Лабораторная работа №2: Параллельное вычисление суммы элементов массива с синхронизацией

«Суммирование элементов с pthread_mutex и pthread_barrier»

Задача: реализовать параллельное вычисление суммы массива с использованием pthread_create и pthread_join, добавить синхронизацию через pthread_mutex_t и pthread_barrier_t.

Что нужно сделать:

- Написать многопоточную программу с синхронизацией;

- Измерить масштабируемость при увеличении числа потоков;

- Сделать выводы о влиянии синхронизации на производительность.

Инструменты: C/C++, POSIX Threads, mutex, barrier, таймеры, профайлеры.

Лабораторная работа №3: Параллельная сортировка массива

«Сортировка с std::thread и std::async»

Задача: реализовать параллельную сортировку массива (merge sort или quick sort) с использованием std::thread и std::async. Измерить время выполнения и сравнить с реализацией на POSIX Threads.

Что нужно сделать:

- Реализовать алгоритм сортировки с использованием потоков C++;

- Сравнить производительность разных подходов;

- Сделать аналитический вывод.

Инструменты: C++, std::thread, std::async, POSIX Threads, таймеры, профайлеры.

Лабораторная работа №4: OpenMP для матричного умножения

«Распараллеливание матричных вычислений»

Задача: использовать директивы #pragma omp parallel for для параллельного умножения матриц. Добавить reduction и schedule(dynamic); сравнить производительность разных стратегий планирования.

Что нужно сделать:

- Реализовать распараллеливание с OpenMP;

- Измерить скорость выполнения;

- Проанализировать влияние стратегий планирования.

Инструменты: C/C++, OpenMP, таймеры, профайлеры.

Лабораторная работа №5: Векторизация и SIMD-вычисления

«Векторное умножение и сложение массивов»

Задача: использовать intrinsic-функции _mm256_add_ps, _mm256_mul_ps, _mm256_fmadd_ps для реализации векторных операций. Сравнить производительность с обычным циклом и проверить влияние выравнивания памяти.

Что нужно сделать:

- Написать программу с SIMD-инструкциями;

- Провести сравнение с обычной реализацией;

- Сделать вывод о влиянии выравнивания памяти.

Инструменты: C/C++, AVX, SSE, таймеры, компилятор с поддержкой SIMD.

Лабораторная работа №6: CUDA-программирование

«Умножение векторов и матриц на GPU»

Задача: реализовать CUDA-ядро для умножения векторов и матриц. Настроить конфигурацию сетки (<<<grid, block>>>), использовать глобальную память, измерить ускорение относительно CPU.

Что нужно сделать:

- Создать CUDA-ядро и хост-программу;

- Измерить время выполнения и ускорение;

- Проанализировать производительность.

Инструменты: CUDA, nvcc, Nsight Systems, таймеры, GPU.

Лабораторная работа №7: OpenCL и кроссплатформенные вычисления

«Векторное сложение на CPU и GPU»

Задача: реализовать программу на OpenCL: создать контекст, очередь команд, компилировать ядро. Сравнить время выполнения с CUDA, протестировать на CPU и GPU.

Что нужно сделать:

- Написать OpenCL-программу;

- Провести тестирование на разных устройствах;

- Сравнить производительность с CUDA.

Инструменты: OpenCL, C/C++, GPU/CPU, таймеры, профайлеры.

Лабораторная работа №8: Оптимизация и профилирование параллельных приложений

«Анализ и ускорение наиболее ресурсоёмких частей программы»

Задача: использовать perf, valgrind, nvprof, Nsight Systems для выявления участков кода, которые потребляют наибольшее время выполнения. Оптимизировать вычисления: улучшить работу с кэшем, подобрать оптимальный размер блоков, уменьшить лишнюю синхронизацию. Составить отчет о достигнутом ускорении и эффективности.

Что нужно сделать:

- Провести профилирование программ и определить участки кода, где программа работает медленнее всего;

- Оптимизировать эти участки в соответствии с рекомендациями профайлера;

- Составить отчет с измерениями ускорения и анализом эффективности.

Инструменты: perf, valgrind, nvprof, Nsight Systems, C/C++, OpenMP, CUDA.

Критерии оценки лабораторных работ:

Отлично: Полное выполнение всех шагов, объяснение всех этапов, ответы на вопросы.

Хорошо: Полное выполнение всех шагов, объяснение всех этапов, частичные ответы на вопросы.

Удовлетворительно: Полное выполнение всех шагов, объяснение всех этапов.

Неудовлетворительно: Невыполнение заданий по ТЗ, неспособность объяснить написанные решения.

Подготовка компьютерных классов с C/C++ (MS Visual Studio, CMake), OpenMP, POSIX Threads, SIMD, CUDA/OpenCL, профилированием и NVIDIA GPU.

Проверка и оценка работ

1. Ручная проверка:
   Общение со студентами по итогам ЛР; вопросы по коду и теоретические вопросы по методам и технологиям компьютерного зрения.
2. Обратная связь:

Разбор ошибок и рекомендации по улучшению

Зачетные проекты на основе кейсов партнеров

Кейсы ПАО «Сбербанк» используемые в дисциплине

Генерация synthetic data для банковских моделей
Описание:

Модели в Сбере требуют большого объёма транзакционных и клиентских данных, которые нельзя использовать напрямую из-за требований ЦБ и ФЗ-152. Задача — разработать метод генерации синтетических банковских данных, максимально близких к реальным по распределениям и поведению.

Зачетный проект на основе кейса:

Высокопроизводительная генерация синтетических финансовых данных

Описание проекта:
Требуется разработать высокопроизводительную систему генерации синтетических банковских данных (транзакции, клиентские профили), которые статистически неотличимы от реальных, но не содержат персональной информации. Из-за огромных объемов данных (миллиарды записей) генерация должна быть максимально оптимизирована и использовать все доступные вычислительные ресурсы CPU и GPU.

Задачи в контексте дисциплины:

1. Параллелизация на CPU: использовать std::thread или OpenMP для параллельного генерирования независимых блоков данных (например, транзакций разных клиентов).

2. Векторизация (SIMD): применить инструкции SSE/AVX для ускорения математических операций при генерации числовых данных (суммы транзакций, проценты, временные метки), которые следуют определенным распределениям (нормальное, пуассона).

3. GPU-ускорение: реализовать на CUDA или OpenCL ядра для массовой параллельной генерации наиболее ресурсоемких частей данных, например, создания взаимосвязанных транзакционных сетей или применения сложных стохастических моделей.

4. Оптимизация и профилирование: с помощью perf, nvprof или Nsight Systems выявить "узкие места", оптимизировать работу с памятью (кэш-локальность) и обеспечить балансировку нагрузки между ядрами CPU и GPU.

Цель проекта: продемонстрировать умение создавать гибридные (CPU+GPU) приложения, способные решать ресурсоемкие задачи в сжатые сроки за счет эффективного использования параллелизма.

Кейсы от «АВАЛАБ» используемые в дисциплине

Анализ обращений клиентов и CRM-переписки
Описание:

В службе клиентского сервиса застройщика ежедневно обрабатываются десятки обращений (e-mail, звонки, мессенджеры). Требуется реализовать систему семантического анализа и классификации NLU: выявлять суть обращений, уровень удовлетворенности, отслеживать повторяющиеся запросы.

Зачетный проект на основе кейса:

Параллельная обработка и семантический анализ потоков CRM-обращений

Параллельная обработка и семантический анализ потоков CRM-обращений

Описание проекта:
Необходимо реализовать ядро системы для предобработки и анализа большого потока текстовых обращений клиентов (электронные письма, чаты). Задача включает этапы, которые можно распараллелить: кодирование текста в векторы, извлечение признаков, начальная классификация. Низкая задержка обработки критична для оперативного реагирования.

Задачи в контексте дисциплины:

1. Многопоточная конвейеризация (Producer-Consumer): создать конвейер обработки данных с использованием std::thread, std::mutex и очередей сообщений. Один поток загружает данные, другой выполняет предобработку (токенизацию), несколько потоков параллельно вычисляют эмбеддинги, а итоговый поток агрегирует результаты.

2. Векторизация вычислений: использовать SIMD-инструкции (AVX) для ускорения линейных алгебраических операций при вычислении базовых текстовых характеристик (например, TF-IDF) или поэлементных операций в небольших нейросетях.

3. GPU-ускорение инференса: реализовать на CUDA или OpenCL вычисления для легковесной нейросетевой модели, выполняющей семантическую классификацию, чтобы обрабатывать сотни обращений одновременно на GPU.

4. Профилирование и оптимизация задержек: использовать valgrind и Nsight Systems для анализа взаимоблокировок (deadlock), "горячих точек" и оптимизации времени отклика системы, минимизируя накладные расходы на синхронизацию.

Цель проекта: показать навыки создания отзывчивых многопоточных систем реального времени, способных эффективно распределять вычислительную нагрузку и использовать низкоуровневую оптимизацию для обработки непрерывных потоков данных.

Зачетно-экзаменационные материалы для промежуточной аттестации (экзамен)

Примеры вопросов для подготовки к зачету

POSIX Threads и многопоточность

1. Что такое поток (thread) и чем он отличается от процесса?
2. Какие преимущества даёт многопоточность в современных приложениях?
3. Что такое контекстное переключение и как оно влияет на производительность?
4. Как создаются потоки с помощью POSIX Threads (pthread_create)?
5. Как завершить поток и вернуть результат (pthread_join)?
6. Что такое mutex и для чего он используется?
7. Какие есть способы синхронизации потоков в POSIX Threads?
8. Что такое условная переменная (pthread_cond_t) и как она работает?
9. Для чего используется барьер (pthread_barrier_t) и как он применяется?
10. Чем отличается конкурентный доступ к памяти от параллельного?
11. Что такое race condition и как его предотвращают?
12. Что такое deadlock и какие способы его избегания?
13. Как измерить производительность многопоточной программы на CPU?
14. Какие проблемы могут возникнуть при увеличении числа потоков?

C/C++ и низкоуровневое программирование

15. Какие особенности C/C++ делают их подходящими для низкоуровневого параллельного программирования?
16. Какие стандартные средства C++ позволяют работать с потоками?
17. Чем отличается std::thread от POSIX Threads?
18. Как работают std::mutex и std::lock_guard?
19. Что такое std::future и std::async?
20. Какие проблемы возникают при использовании глобальных переменных в многопоточном коде?
21. Что такое атомарные операции (std::atomic) и зачем они нужны?
22. Как реализовать безопасный доступ к разделяемым ресурсам в C++?

OpenMP

23. Что такое OpenMP и для чего он используется?
24. Как включить OpenMP в проект на C/C++?
25. Что делает директива #pragma omp parallel?
26. Чем отличается parallel for от обычного for?

27. Что такое reduction в OpenMP и как она работает?
28. Какие типы schedule существуют в OpenMP (static, dynamic, guided)?
29. Для чего используются критические секции (#pragma omp critical)?
30. Как измерить производительность OpenMP-программ?
31. Какие проблемы возникают при неправильном использовании OpenMP?

Векторизация и SIMD (SSE/AVX)

32. Что такое SIMD и чем оно отличается от MIMD?
33. Какие преимущества даёт использование SSE/AVX инструкций?
34. Как реализовать векторные операции с помощью _mm256_add_ps или _mm256_mul_ps?
35. Чем отличается 128-битная SSE от 256-битной AVX?
36. Как влияет выравнивание памяти на производительность SIMD-операций?
37. Какие ограничения накладывает использование SIMD на алгоритмы?
38. Как измерять ускорение при векторизации кода?

GPU-программирование (CUDA, ROCm, OpenCL)

39. Что такое CUDA и для чего она используется?
40. Как устроена модель потоков и блоков в CUDA?
41. Чем отличается глобальная, локальная и shared память в CUDA?
42. Что такое OpenCL и чем он отличается от CUDA?
43. Какие шаги необходимы для выполнения OpenCL-ядра на GPU?
44. Какие методы профилирования применяются для GPU-программ и что они измеряют?

Перечень компетенций (части компетенции), проверяемых оценочным средством - ОПК-3; PL-1; PL-2; PL-3

Практические задания к зачету

Задание 1. Создание потоков POSIX

- Реализовать программу, создающую несколько потоков для параллельного вычисления суммы и среднего элементов массива.
- Измерить ускорение при увеличении числа потоков.

Задание 2. Синхронизация с mutex и barrier

- Использовать pthread_mutex_t и pthread_barrier_t для защиты разделяемого ресурса и синхронизации потоков.
- Измерить масштабируемость решения при разном количестве потоков.

Задание 3. Параллельная сортировка на C++

- Реализовать merge sort или quick sort с использованием std::thread и std::async.
- Сравнить время выполнения с реализацией на POSIX Threads.

Задание 4. Распараллеливание с OpenMP

- Использовать директиву #pragma omp parallel for для умножения матриц.
- Добавить reduction и изменить стратегию schedule. Сравнить производительность.

Задание 5. Векторизация SIMD

- Реализовать сложение и умножение массивов с помощью SSE/AVX (_mm256_add_ps, _mm256_mul_ps).
- Сравнить время выполнения с обычным циклом.

Задание 6. Оптимизация векторных операций

- Проверить влияние выравнивания данных в памяти на производительность SIMD.
- Измерить ускорение на разных массивах.

Задание 7. CUDA: умножение векторов

- Реализовать CUDA-ядро для умножения двух векторов.
- Настроить конфигурацию блоков и сетки, измерить ускорение относительно CPU.

Задание 8. CUDA: умножение матриц

- Реализовать умножение матриц на GPU с использованием shared памяти для ускорения.
- Провести сравнение с CPU-реализацией.

Задание 9. OpenCL: векторное сложение

- Написать программу на OpenCL для сложения массивов.
- Сравнить производительность с CUDA-реализацией и CPU.

Задание 10. ROCm/HIP

- Реализовать простое GPU-ядро на ROCm/HIP для векторного умножения.
- Сравнить результаты с CUDA.

Задание 11. Профилирование CPU

- Использовать perf или Valgrind для анализа горячих участков многопоточной программы на CPU.
- Оптимизировать вычисления и измерить ускорение.

Задание 12. Профилирование GPU

- Использовать Nsight Systems или nvprof для анализа CUDA-ядра.
- Оптимизировать размер блоков и использование памяти для ускорения.

Задание 13. Комбинированная оптимизация

- Реализовать программу, комбинирующую многопоточность, SIMD и GPU-вычисления (например, суммирование больших массивов).
- Провести измерение ускорения на всех уровнях и подготовить сравнительный отчет.

Перечень компетенций (части компетенции), проверяемых оценочным средством - ОПК-3; PL-1; PL-2; PL-3

4.2 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Методические рекомендации, определяющие процедуры оценивания на зачете:

Процедура промежуточной аттестации проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

Итоговой формой контроля сформированности компетенций у обучающихся по дисциплине является зачет. Студенты обязаны сдать зачет в соответствии с расписанием и учебным планом.

ФОС промежуточной аттестации состоит из вопросов к зачету и результатов текущего контроля.

Зачет по дисциплине преследует цель оценить работу студента за курс, получение теоретических знаний, их прочность, развитие творческого мышления, приобретение навыков самостоятельной работы, умение применять полученные знания для решения практических задач.

Форма проведения зачета: устно.

Результат сдачи зачета заноситься преподавателем в зачетную ведомость и зачетную книжку.

Оценивание уровня освоения дисциплины основывается на качестве выполнения студентом заданий текущего контроля и ответов на вопросы зачета.

Критерии оценки:

1. Оценка ответов на вопросы к зачету (50% итоговой оценки)

Зачет

Полные, развернутые ответы с демонстрацией глубокого понимания темы.
Ответы содержат основные идеи, но без углубленного анализа.
Использование примеров, формул, корректных терминов.
Возможны небольшие ошибки в деталях или формулировках.
Умение анализировать и сравнивать методы.
% выполнения: 60–100% (допускаются незначительные неточности).

Незачет

Отсутствие понимания ключевых концепций.
Грубые ошибки или неспособность ответить на большую часть вопросов.
% выполнения: <60%.

2. Оценка выполнения практических кейсов и лабораторных работ (50% итоговой оценки)

Зачет

Полное выполнение всех этапов кейса с инновационными решениями.
Четкая документация кода и анализ результатов.
% выполнения: 60–100%.

или

Выполнены основные задачи, но без дополнительной оптимизации.

или

Решены базовые задачи, но с критическими ошибками.
Низкое качество кода или отсутствие анализа.

Незачет

Невыполнение ключевых этапов.
Код нерабочий или отсутствует.
% выполнения: <60%.

Методические рекомендации, определяющие процедуры оценивания лабораторных работ:

Процедура оценивания лабораторных работ проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

По каждой лабораторной работе оформляется отчет. Отчеты сдаются на проверку руководителю в течение курса по мере их выполнения, и защищаются студентами в установленном порядке.

При защите отчета студенту могут быть заданы вопросы и дополнительные задания по сути лабораторной работы, в том числе из списка контрольных вопросов к данной лабораторной работе. При неудовлетворительной оценке знаний студента по теме данного отчета, студент возвращается к повторному изучению соответствующих материалов, после чего допускается к повторной защите. Неудовлетворительно выполненный отчет также возвращается на доработку.

Отчет должен содержать заголовок, тему лабораторной работы, цель, задание, индивидуальную тему, описание хода выполнения работы, необходимые прикладные материалы (схемы, макеты документов и т.п.), в соответствии с требованиями к содержанию, и выводы по работе.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на зачете;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4.3. Методические указания по организации вычислительной инфраструктуры

Требования к аппаратному и программному обеспечению рабочих мест

Аппаратные требования.

Для выполнения лабораторных работ по параллельному и низкоуровневому программированию студентам и преподавателю необходим стационарный компьютер или ноутбук с современной конфигурацией. Рекомендуется многопроцессорный CPU, например Intel Core i3/i5/i7 не ниже 4-го поколения или аналогичный AMD Ryzen, с поддержкой многопоточности и оперативной памятью не менее 8 ГБ. Для работы с GPU-

вычислениями требуется видеокарта NVIDIA для CUDA или совместимая с OpenCL/ROCm. Компьютер должен иметь стабильное подключение к сети Интернет со скоростью не ниже 5–10 Мбит/с для скачивания SDK, библиотек и обновлений программного обеспечения.

Программные требования.

На рабочих станциях должна быть установлена современная операционная система, включая Windows 10 или 11, актуальные версии macOS или дистрибутивы GNU/Linux, при этом системы должны регулярно обновляться для поддержания безопасности и совместимости с инструментами курса. Для разработки необходим современный компилятор C/C++ с поддержкой CMake, например MS Visual Studio на Windows или GCC/Clang на Linux/macOS, а также библиотеки и инструменты для параллельного программирования, такие как POSIX Threads, OpenMP, SIMD-интринсики для SSE/AVX, CUDA Toolkit для NVIDIA GPU, OpenCL SDK и ROCm/HIP для совместимых устройств. Разработка рекомендуется через интегрированную среду или текстовый редактор с поддержкой C/C++, например Visual Studio, VS Code или CLion. Для анализа и оптимизации производительности необходимо использовать инструменты профилирования CPU и GPU, включая perf, Valgrind (через WSL на Windows), Nsight Systems, nvprof для CUDA и средства профилирования OpenCL/ROCm.

Дополнительное программное обеспечение.

Студенты также должны иметь доступ к системе контроля версий Git, интерпретатору Python версии 3.10 и выше с менеджером пакетов pip или conda для анализа результатов и построения графиков, при необходимости с установкой Jupyter Notebook/Lab. Для локального тестирования и отладки программ может использоваться Docker, при этом на Windows требуется Docker Desktop с WSL2, а на Linux и macOS платформа поддерживается напрямую. Все программное обеспечение должно быть настроено так, чтобы студенты имели доступ ко всем инструментам во время лабораторных работ, а преподаватель мог управлять инфраструктурой и контролировать результаты, включая репозитории, CI/CD и тестирование. Необходимо обеспечить разрешение исходящих подключений по HTTPS, открытые порты 80 и 443, а также наличие прав на установку программного обеспечения или взаимодействие с системным администратором для их установки.

План-проспект методических указаний по организации лабораторных работ по дисциплине " Параллельное и низкоуровневое программирование"

1. Общие сведения

Дисциплина: «Параллельное и низкоуровневое программирование»

Вид обеспечения: проведение лабораторных работ

Условия применения: курс рассчитан на студентов 2–3-го года обучения; наличие доступа к вычислительным ресурсам CPU и GPU; использование открытых библиотек и SDK для C/C++, OpenMP, POSIX Threads, CUDA, OpenCL/ROCm; доступ к примерам и исходным данным для лабораторных работ.

2. Цели, задачи и ожидаемые результаты

Цели: закрепление теоретических знаний по параллельному и низкоуровневому программированию на практике; развитие навыков проектирования, реализации и оптимизации многопоточных и векторных вычислений; подготовка к решению задач на CPU и GPU в индустриальных проектах.

Задачи: обеспечить студентов структурированными лабораторными работами; предоставить доступ к необходимым вычислительным ресурсам; организовать проверку и обратную связь по выполненным работам.

Ожидаемые результаты: умение создавать и оптимизировать многопоточные приложения на C/C++; навыки работы с OpenMP, POSIX Threads и SIMD-интринсиками (SSE/AVX); опыт реализации GPU-вычислений на CUDA, OpenCL и ROCm; способность анализировать производительность и масштабируемость решений.

3. Порядок реализации

3.1. Задача №1: Подготовка лабораторных работ

Определение тем лабораторных работ: введение в многопоточность и POSIX Threads; параллельные вычисления на C++ (std::thread, std::async); распараллеливание с OpenMP; векторизация и SIMD (SSE/AVX); GPU-программирование с CUDA, OpenCL и ROCm; оптимизация и профилирование CPU и GPU-приложений; комбинированные задачи на CPU и GPU; защита курсовых проектов.

Разработка заданий включает пошаговые инструкции, примеры кода и контрольные вопросы.

Пример разработки задания для лабораторной работы «Многопоточные вычисления на CPU»

Название: «Суммирование элементов массива с использованием POSIX Threads и std::thread»

Цель: научиться создавать многопоточные программы на C/C++ и анализировать их производительность.

Задачи: реализовать программу для параллельного суммирования массива с использованием pthread_create и pthread_join; добавить синхронизацию через pthread_mutex_t и pthread_barrier_t; сравнить производительность реализации на POSIX Threads и std::thread; измерить ускорение при увеличении числа потоков.

Ожидаемые результаты: умение создавать и синхронизировать потоки на CPU; понимание влияния синхронизации и числа потоков на производительность; навыки сравнения разных подходов к параллельным вычислениям.

Инструменты и библиотеки: C/C++, POSIX Threads, std::thread, std::mutex, std::future, OpenMP, таймеры, компиляторы GCC/Clang или MS Visual Studio, профайлеры (perf, Valgrind).

Исходные данные: массивы чисел размером 10^6–10^8 элементов.

Ход работы: определить количество потоков; реализовать параллельное суммирование с синхронизацией; измерить время выполнения; оформить таблицу с результатами; проанализировать масштабируемость.

Требования к отчёту: код программы; таблица с измерениями времени; графики ускорения и масштабируемости; аналитическая часть с выводами.

Критерии оценки (зачтено/незачтено): зачет ставится, если программа корректно реализована с синхронизацией, измерены времена выполнения, оформлен отчет с анализом; незачет при отсутствии синхронизации, неполных измерениях или отсутствии анализа.

Формируемые компетенции:

- ОПК-3 Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности

- PL-1 (Б) Способен применять язык программирования Python для решения задач в области ИИ

- PL-2 (П) Способен применять JVM-совместимые языки программирования для решения задач в области ИИ

- PL-3 (Б) Способен применять языки программирования C/C++ для решения задач в области ИИ

5. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)

5.1 Основная литература:

1. Богачёв К.Ю. Основы параллельного программирования: учебное пособие. – М.:Лаборатория знаний, 2020. – 345 с.

2. Лаптев В.В. C++ Объектно-ориентированное программирование: Учебное пособие. – СПб.: Питер, 2008 – 464 с.

3. Алексеев, В.Е. Структуры данных. Модели вычислений / В.Е. Алексеев, В.А. Таланов. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 248 с. : схем., ил. - (Основы информационных технологий)

4. Эндрюс Г.Р. Основы многопоточного, параллельного программирования. М.:Издательский дом “Вильямс”. 2003. 512с

5. Малявко А.А. Параллельное программирование на основе технологий OpenMP, CUDA, OpenCL, MPI: учебное пособие для вузов / А. А. Малявко. - 3-е изд., испр. и доп. - Москва : Юрайт, 2022. - 135 с. - URL: https://urait.ru/bcode/492127 (дата обращения: 29.11.2022). - Режим доступа: для авториз. пользователей. - ISBN 978-5-534-14116-0. - Текст: электронный.

5.2 Дополнительная литература:

1. Биллиг В.А. Параллельные вычисления и многопоточное программирование. – М.: Национальный Открытый Университет «ИНТУИТ», 2016. – 311 с. – URL: https://biblioclub.ru/index.php?page=book&id=428948.

2. Туральчук К.А. Параллельное программирование с помощью языка C#. – М.: Национальный Открытый Университет «ИНТУИТ», 2016. – 190 с. –URL: https://biblioclub.ru/index.php?page=book&id=429098.

5.3. Периодические издания:

1. Базы данных компании «Ист Вью» http://dlib.eastview.com
2. Электронная библиотека GREBENNIKON.RU https://grebennikon.ru/

5.4. Интернет-ресурсы, в том числе современные профессиональные базы данных и информационные справочные системы

Электронно-библиотечные системы (ЭБС):

1. ЭБС «ЮРАЙТ» https://urait.ru/
2. ЭБС «УНИВЕРСИТЕТСКАЯ БИБЛИОТЕКА ОНЛАЙН» http://www.biblioclub.ru/
3. ЭБС «BOOK.ru» https://www.book.ru
4. ЭБС «ZNANIUM.COM» www.znanium.com
5. ЭБС «ЛАНЬ» https://e.lanbook.com

Профессиональные базы данных

1. Scopus http://www.scopus.com/
2. ScienceDirect https://www.sciencedirect.com/
3. Журналы издательства Wiley https://onlinelibrary.wiley.com/
4. Научная электронная библиотека (НЭБ) http://www.elibrary.ru/
5. Полнотекстовые архивы ведущих западных научных журналов на Российской платформе научных журналов НЭИКОН http://archive.neicon.ru

6. Национальная электронная библиотека (доступ к Электронной библиотеке диссертаций Российской государственной библиотеки (РГБ) https://rusneb.ru/
7. Президентская библиотека им. Б.Н. Ельцина https://www.prlib.ru/
8. База данных CSD Кембриджского центра кристаллографических данных (CCDC) https://www.ccdc.cam.ac.uk/structures/
9. Springer Journals: https://link.springer.com/
10. Springer Journals Archive: https://link.springer.com/
11. Nature Journals: https://www.nature.com/
12. Springer Nature Protocols and Methods: https://experiments.springernature.com/sources/springer-protocols
13. Springer Materials: http://materials.springer.com/
14. Nano Database: https://nano.nature.com/
15. Springer eBooks (i.e. 2020 eBook collections): https://link.springer.com/
16. "Лекториум ТВ" http://www.lektorium.tv/
17. Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Бесплатные образовательные ресурсы

1. Jupyter Notebook – интерактивные вычисления
2. Visual Studio Code – редактор кода с поддержкой Python
3. Google Scholar/arXiv – доступ к научным публикациям

Ресурсы свободного доступа

1. КиберЛенинка http://cyberleninka.ru/;
2. Американская патентная база данных http://www.uspto.gov/patft/
3. Министерство науки и высшего образования Российской Федерации https://www.minobrnauki.gov.ru/;
4. Федеральный портал "Российское образование" http://www.edu.ru/;
5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru/;
6. Единая коллекция цифровых образовательных ресурсов http://school-collection.edu.ru/ .
7. Проект Государственного института русского языка имени А.С. Пушкина "Образование на русском" https://pushkininstitute.ru/;
8. Справочно-информационный портал "Русский язык" http://gramota.ru/;
9. Служба тематических толковых словарей http://www.glossary.ru/;
10. Словари и энциклопедии http://dic.academic.ru/;
11. Образовательный портал "Учеба" http://www.ucheba.com/;
12. Законопроект "Об образовании в Российской Федерации". Вопросы и ответы http://xn--273--84d1f.xn--p1ai/voprosy_i_otvety

Собственные электронные образовательные и информационные ресурсы КубГУ

1. Электронный каталог Научной библиотеки КубГУ http://megapro.kubsu.ru/MegaPro/Web
2. Электронная библиотека трудов ученых КубГУ http://megapro.kubsu.ru/MegaPro/UserEntry?Action=ToDb&idb=6
3. Среда модульного динамического обучения http://moodle.kubsu.ru
4. База учебных планов, учебно-методических комплексов, публикаций и конференций http://infoneeds.kubsu.ru/
5. Библиотека информационных ресурсов кафедры информационных образовательных технологий http://mschool.kubsu.ru;
6. Электронный архив документов КубГУ http://docspace.kubsu.ru/
7. Электронные образовательные ресурсы кафедры информационных систем и технологий в образовании КубГУ и научно-методического журнала "ШКОЛЬНЫЕ ГОДЫ" http://icdau.kubsu.ru/

5.5. Публикации конференций A*

1. Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. 2025. Generating Long Semantic IDs in Parallel for Recommendation. In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25). Association for Computing Machinery, New York, NY, USA, 956–966. https://doi.org/10.1145/3711896.3736979
2. Zijian Shi and John Cartlidge. 2022. State Dependent Parallel Neural Hawkes Process for Limit Order Book Event Stream Prediction and Simulation. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22). Association for Computing Machinery, New York, NY, USA, 1607–1615. https://doi.org/10.1145/3534678.3539462
3. B. Zhuang, Q. Wu, C. Shen, I. Reid and A. v. d. Hengel, "Parallel Attention: A Unified Framework for Visual Object Discovery Through Dialogs and Queries," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018, pp. 4252-4261, doi: 10.1109/CVPR.2018.00447.
4. Peng Peng, Shengyi Ji, Zhen Tian, Hongbo Jiang, Weiguo Zheng, and Xuecang Zhang. 2023. Locality Sensitive Hashing for Optimizing Subgraph Query Processing in Parallel Computing Systems. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23). Association for Computing Machinery, New York, NY, USA, 1885–1896. https://doi.org/10.1145/3580305.3599419
5. Hui Sun, Yanfeng Ding, Liping Yi, Huidong Ma, Gang Wang, Xiaoguang Liu, Cheng Zhong, and Wentong Cai. 2025. PMKLC: Parallel Multi-Knowledge Learning-based Lossless Compression for Large-Scale Genomics Database. In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25). Association for Computing Machinery, New York, NY, USA, 2725–2734. https://doi.org/10.1145/3711896.3737083

6. Методические указания для обучающихся по освоению дисциплины (модуля)

По курсу предусмотрено проведение лекционных занятий, на которых дается систематизированный материал по параллельному и низкоуровневому программированию. В ходе лекций рассматриваются ключевые концепции многопоточности, синхронизации потоков, архитектуры современных процессоров, векторизации и GPU-вычислений. Особое внимание уделяется практическим аспектам — разбираются примеры реализации многопоточных и параллельных алгоритмов на C/C++, применение OpenMP и POSIX Threads, использование SIMD-интринсиков (SSE/AVX), GPU-программирование на CUDA, OpenCL и ROCm. На лекциях анализируются типичные ошибки при проектировании параллельных приложений и оптимизации вычислений. После каждой лекции рекомендуется выполнять практические задания для закрепления материала: реализация многопоточных функций, распараллеливание циклов, оптимизация векторных вычислений и GPU-ядр.

Лабораторные занятия курса посвящены практическому освоению методов параллельного и низкоуровневого программирования. На занятиях детально разбираются готовые примеры реализации многопоточных алгоритмов, распараллеливание циклов с

OpenMP, работа с потоками POSIX Threads, оптимизация кода с использованием SIMD-инструкций, организация вычислений на GPU с помощью CUDA, OpenCL и ROCm. Студенты работают с реальными вычислительными задачами, например, суммирование и сортировка массивов, матричные операции, моделирование физических процессов, и применяют инструменты для измерения производительности (perf, Valgrind, Nsight Systems). После каждого лабораторного занятия предлагаются задания для самостоятельного закрепления материала — модификация рассмотренных алгоритмов, тестирование многопоточных реализаций и оптимизация производительности.

При самостоятельной работе студентам необходимо изучать рекомендованную литературу (учебники, статьи, документацию библиотек и SDK) для глубокого понимания теоретических основ параллельного программирования и современных подходов к оптимизации вычислений. Выполняя проектные задания, студент должен уметь: формулировать задачу параллельных вычислений; подбирать и подготавливать данные; выбирать и реализовывать подходящие методы распараллеливания и оптимизации на CPU и GPU; оценивать эффективность и производительность решений с помощью соответствующих инструментов и метрик. Особое внимание уделяется навыкам отладки, профилирования и оптимизации кода.

Важнейшим компонентом курса является самостоятельная проектная работа, в ходе которой студент разрабатывает завершённое программное решение с использованием многопоточности, векторизации и GPU-вычислений (например, ускоренное суммирование больших массивов, параллельная сортировка или вычислительная задача с матрицами). Такой проект позволяет закрепить навыки проектирования и реализации комплексных решений на CPU и GPU, а также оценивать их производительность.

Для студентов с ограниченными возможностями здоровья предусмотрены дополнительные индивидуальные консультации, на которых преподаватель разъясняет сложные аспекты дисциплины, помогает адаптировать задания и обеспечивает специальные условия для освоения практических навыков работы с многопоточными, векторными и GPU-вычислениями. Индивидуальный подход позволяет таким студентам полноценно участвовать в учебном процессе и достигать требуемых результатов обучения.

Кейсы ПАО «Сбербанк» используемые в дисциплине

Генерация synthetic data для банковских моделей
Описание:

Модели в Сбере требуют большого объёма транзакционных и клиентских данных, которые нельзя использовать напрямую из-за требований ЦБ и ФЗ-152. Задача — разработать метод генерации синтетических банковских данных, максимально близких к реальным по распределениям и поведению.

Кейсы от «АВАЛАБ» используемые в дисциплине

Анализ обращений клиентов и CRM-переписки
Описание:

В службе клиентского сервиса застройщика ежедневно обрабатываются десятки обращений (e-mail, звонки, мессенджеры). Требуется реализовать систему семантического анализа и классификации NLU: выявлять суть обращений, уровень удовлетворенности, отслеживать повторяющиеся запросы.

7. Материально-техническое обеспечение по дисциплине (модулю)

Виртуальные машины, кластер Managed Kubernetes и ресурсы GPU в облаке предоставляется индустриальным партнером ПАО «Сбербанк»:

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.

№	Продукт	Параметры продукта	Кол-во	Кол-во конфигураций	Ед. изм.
1	Виртуальная машина	Виртуальная машина 10% vCPU 2 vCPU 4 RAM	1	60	Шт
		ОС Ubuntu 22.04	1		Шт
		Системный диск SSD	1		Шт
			10		Гб
		Аренда публичного IP	1		Шт
2	Виртуальная машина с GPU	Виртуальная машина с GPU NVIDIA® Tesla® V100 2 GPU 8 vCPU 128 ГБ RAM	1	1	Шт
		ОС Ubuntu_24.04	1		Шт
		Системный диск SSD	1		Шт
			2000		Гб
		Диск SSD	1		Шт
			4096		Гб
		Диск SSD	1		Шт
			4096		Гб
		Аренда публичного IP	1		Шт

№	Вид работ	Наименование учебной аудитории, ее оснащенность оборудованием и техническими средствами обучения
1	Лекционные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
2	Лабораторные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, проектором, программным обеспечением
3	Практические занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
4	Групповые (индивидуальные) консультации	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением

5	Текущий контроль, промежуточная аттестация	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
6	Самостоятельная работа	Кабинет для самостоятельной работы, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», программой экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета.