МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Институт кибербезопасности и цифровых технологий

Региональный партнёр
ФГБОУ ВО
«Дагестанский государственный технический университет»

УТВЕРЖДАЮ
И.о. ректора ФГБОУ ВО
Н.Л. Баламирзоев
«09» 2023 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Б1.В.ДВ.02.02 «Параллельное программирование»

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Направленность (профиль подготовки): «Прикладной искусственный интеллект»

Квалификация выпускника бакалавр

Форма обучения очная

Махачкала 2023

1. ПАСПОРТ
фонда оценочных средств

по дисциплине Б1.В.ДВ.02.02 «Параллельное программирование»
1.1. Результаты обучения по дисциплине:

1.1. Результаты обучения по дисциплине:

Код компетенции	Наименование компетенции	Индикатор достижения компетенции	В результате освоения дисциплины обучающийся должен:	Другая дисциплина (дисциплины) / практика, участвующая в формировании компетенции
ОПК-4	Способен разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	**ОПК-4.1 (Адаптированный)** Применяет технологии параллельного программирования (OpenMP, MPI) для разработки компонентов вычислительных систем.	**Знать:** модели параллельного программирования (разделяемая и распределенная память), синтаксис и семантику основных директив OpenMP и функций MPI. **Уметь:** разрабатывать корректно работающие параллельные программы для систем с разделяемой (OpenMP) и распределенной (MPI) памятью. **Владеть:** навыками отладки, тестирования и оценки эффективности параллельных приложений.	Архитектура ЭВМ и систем. Программирование. Операционные системы.
		ОПК-4.2 (Адаптированный) Использует тех-	**Знать:** архитектурные особенности GPU, модель выполнения	Алгоритмы и структуры данных. Математи-

Код компетенции	Наименование компетенции	Индикатор достижения компетенции	В результате освоения дисциплины обучающийся должен:	Другая дисциплина (дисциплины) / практика, участвующая в формировании компетенции
		нологии GPU-программирования (CUDA) для ускорения вычислительных задач.	CUDA, иерархию памяти. **Уметь:** разрабатывать и оптимизировать CUDA-ядра для типовых вычислительных операций. **Владеть:** инструментарием CUDA Toolkit для разработки и профилирования.	ческие основы искусственного интеллекта.
ПК-1	Способен проводить предпроектное обследование, анализ предметной области, проектировать и моделировать компоненты информационных систем	**ПК-1.1 (Адаптированный)** Анализирует алгоритмы и вычислительные задачи на предмет возможности и целесообразности распараллеливания.	**Знать:** метрики параллельной эффективности (ускорение, масштабируемость), закон Амдала, принципы декомпозиции данных и задач. **Уметь:** выявлять независимые ветви вычислений в алгоритме, оценивать потенциальный выигрыш от параллелизации. **Владеть:** методами проектирования параллельных алгоритмов.	Теория алгоритмов. Методы оптимизации. Системное программирование.
		ПК-1.2 (Адаптированный) Проектирует архитектуру параллельного приложения для решения прикладных задач, в том числе в области ИИ.	**Знать:** типовые параллельные паттерны (Map, Reduce, Fork-Join, Pipeline), стратегии распараллеливания алгоритмов линейной алгебры и машинного обучения. **Уметь:** выбирать подходящую модель программирования (OpenMP, MPI, CUDA) и проектировать схему взаимодействия	Системы искусственного интеллекта. Нейронные сети. Базы данных.

Код компетенции	Наименование компетенции	Индикатор достижения компетенции	В результате освоения дисциплины обучающийся должен:	Другая дисциплина (дисциплины) / практика, участвующая в формировании компетенции
			параллельных компонентов. **Владеть:** основами проектной документации для параллельных систем.	
ПК-9	Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем	**ПК-9.1 (Адаптированный)** Настраивает среду разработки и выполнения для параллельного программирования.	**Знать:** состав и требования к программному стеку для разработки параллельных приложений (компиляторы, библиотеки, драйверы). **Уметь:** устанавливать и конфигурировать MPI-среду (OpenMPI/MPICH), CUDA Toolkit, настраивать компиляторы с поддержкой OpenMP. **Владеть:** навыками работы в командной строке Linux, написания сборочных скриптов (Makefile, CMake).	Администрирование информационных систем. Программная инженерия.
		ПК-9.2 (Адаптированный) Оценивает аппаратные требования и ограничения для выполнения параллельных программ.	**Знать:** характеристики современных многоядерных CPU, GPU, топологию межпроцессорных соединений, влияние памяти и кэша на производительность. **Уметь:** оценивать необходимые вычислительные ресурсы (количество ядер/потоков, объем ОЗУ и видеопамяти) для конкретной параллельной задачи. **Владеть:** чтением и интерпретацией	Архитектура ЭВМ и систем. Сети и телекоммуникации.

Код компетенции	Наименование компетенции	Индикатор достижения компетенции	В результате освоения дисциплины обучающийся должен:	Другая дисциплина (дисциплины) / практика, участвующая в формировании компетенции
			спецификаций аппаратного обеспечения.	

1.2. Программа оценивания контролируемой компетенции:

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции / индикатора	Наименование оценочного средства
1.	Введение в параллельное программирование. Модели и паттерны.	ОПК-4/ОПК-4.1 ПК-1/ПК-1.1	Контрольный тест (Блок А). Собеседование по теоретическим основам.
2.	Параллельное программирование на разделяемой памяти (OpenMP).	ОПК-4/ОПК-4.1 ПК-9/ПК-9.1	Защита лабораторных работ №1-3. Контрольный тест (Блок Б).
3.	Программирование на распределенной памяти (MPI).	ОПК-4/ОПК-4.1 ПК-1/ПК-1.2	Защита лабораторных работ №4-5. Контрольный тест (Блок В).
4.	Основы программирования GPU (CUDA).	ОПК-4/ОПК-4.2 ПК-9/ПК-9.2	Защита лабораторных работ №6-7. Контрольный тест (Блок Г).
5.	Параллельные алгоритмы для задач ИИ. Гибридное программирование.	ПК-1/ПК-1.2 ОПК-4/ОПК-4.2	Защита лабораторной работы №8. Контрольный тест (Блок Д). Защита проекта/отчета по СРС.
Форма промежуточной аттестации в 8 семестре — зачет.			

2. КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

2.1. Вопросы к зачету по дисциплине «Параллельное программирование»

1. Понятия параллелизма, конкуренции, процессов и потоков. Аппаратные предпосылки параллельных вычислений.
2. Классификация параллельных архитектур (Флинн). Характеристики современных многоядерных и многосокетных систем.
3. Модели параллельного программирования: разделяемая память, передача сообщений, гибридная модель.
4. Метрики оценки эффективности параллельных программ: ускорение, эффективность, масштабируемость.
5. Закон Амдала и закон Густафсона. Практические следствия.
6. Архитектура OpenMP. Модель fork-join. Основные директивы и клаузы.
7. Распределение работы в OpenMP: директива for, клауза schedule. Директива sections.
8. Управление областью видимости данных в OpenMP: клаузы shared, private, firstprivate, lastprivate.
9. Синхронизация в OpenMP: критические секции (critical), атомарные операции (atomic), барьеры, явные блокировки.
10. Редукционные операции в OpenMP. Модель задач (tasks).
11. Стандарт MPI: назначение, основные понятия (коммуникатор, ранг, размер).
12. Блокирующие точечные операции в MPI: MPI_Send, MPI_Recv. Соответствие сообщений.
13. Коллективные операции связи в MPI: MPI_Bcast, MPI_Scatter, MPI_Gather, MPI_Allgather.
14. Коллективные операции редукции в MPI: MPI_Reduce, MPI_Allreduce.
15. Архитектура GPU NVIDIA. Иерархия потоков CUDA: grid, block, thread. Понятие warp.
16. Модель памяти CUDA: регистры, разделяемая, глобальная, константная, текстурная память.
17. Написание и запуск CUDA-ядер. Спецификаторы функций __global__, __device__, __host__.
18. Основные принципы оптимизации CUDA-программ: объединенный доступ к памяти, использование разделяемой памяти.
19. Параллельные паттерны: Map, Reduce, Stencil. Их реализация с использованием OpenMP, MPI, CUDA.
20. Стратегии распараллеливания алгоритмов машинного обучения: Data Parallelism, Model Parallelism.

21. Гибридное программирование MPI+OpenMP: преимущества, области применения, типовые схемы.
22. Проблемы параллельного программирования: гонки данных, взаимные блокировки (deadlock), ложное разделение кэша (false sharing).
23. Инструменты отладки и профилирования параллельных приложений (например, gdb, Intel VTune, NVIDIA Nsight).
24. Принципы проектирования параллельных программ: декомпозиция, балансировка нагрузки, минимизация накладных расходов.

2.2. Вопросы для проверки уровня обученности (ЗНАТЬ и УМЕТЬ)

1. В чем разница между параллелизмом и конкуренцией?
2. Приведите пример задачи, где ускорение будет ограничено законом Амдала.
3. Как директива `#pragma omp parallel for` распределяет итерации цикла между потоками по умолчанию?
4. Для чего используется клауза `reduction(+:sum)` в OpenMP?
5. Чем отличается `MPI_Send` от `MPI_Isend`?
6. Какой коллективной операцией MPI эффективнее всего реализовать поэлементное суммирование распределенных массивов?
7. Сколько нитей будет в блоке с размерностью `dim3 blockDim(8, 4, 2)`?
8. Какую функцию CUDA API используют для освобождения памяти на устройстве?
9. В чем основное преимущество разделяемой памяти (`__shared__`) перед глобальной в CUDA?
10. Какой параллельный паттерн лежит в основе операции скалярного произведения векторов?
11. Как можно избежать состояния гонки при инкременте общей переменной в OpenMP?
12. Что такое коммуникатор `MPI_COMM_WORLD`?
13. Как в гибридной программе (MPI+OpenMP) обычно соотносятся MPI-процессы и OpenMP-потоки на кластере?
14. Для чего используется директива `#pragma omp barrier`?
15. Какие факторы могут ограничить масштабируемость MPI-программы при увеличении числа процессов?

3. ЗАДАНИЯ ДЛЯ ПРОВЕРКИ ОБУЧЕННОСТИ (ВЛАДЕТЬ)

Задание 1. Анализ и распараллеливание алгоритма (OpenMP)

Проанализируйте приведенный последовательный алгоритм вычисления числа π методом численного интегрирования. Определите, какие части алгоритма можно распараллелить. Реализуйте параллельную версию с использованием OpenMP, обеспечив корректную синхронизацию и минимизацию накладных расходов. Измерьте ускорение на 2, 4, 8 потоках. Объясните полученные результаты.

```
cdouble pi_sequential(long n) {
   double h = 1.0 / (double) n;
   double sum = 0.0;
   for (long i = 0; i < n; i++) {
      double x = h  ((double)i + 0.5);
      sum += 4.0 / (1.0 + xx);
   }
   return h  sum;}
```

Указания к выполнению: Обратите внимание на редукцию переменной sum. Исследуйте влияние различных стратегий планирования (schedule) на равномерность загрузки потоков.

Задание 2. Распределенная обработка данных (MPI)

Разработайте MPI-программу для параллельного чтения большого текстового файла, подсчета частоты встречаемости слов и вывода K самых частых слов. Исходный файл должен быть равномерно разделен между процессами. Используйте коллективные операции для сбора и объединения результатов. Сравните производительность с последовательной версией.

Указания к выполнению: Используйте MPI_File_read_at для независимого чтения каждым процессом своей части файла. Для сбора локальных словарей в глобальный используйте операции MPI_Gather или MPI_Allgather с последующим объединением на одном процессе. Подумайте об эффективности передачи хэш-таблиц.

Задание 3. Реализация и оптимизация CUDA-ядра

Реализуйте на CUDA ядро для умножения разреженной матрицы, хранящейся в формате CSR (Compressed Sparse Row), на плотный вектор (SpMV). Проведите базовую оптимизацию: организуйте объединенный доступ к векторам, используйте разделяемую память для кэширования часто используемых данных. Сравните производительность с аналогичной ре-

ализацией на CPU (последовательной и OpenMP).

Указания к выполнению: Один из подходов — назначить по одной строке матрицы на блок нитей. Разделяемую память можно использовать для кэширования элементов вектора, к которым идет многократное обращение в рамках блока.

Задание 4. Проектирование гибридного приложения

Спроектируйте архитектуру гибридного приложения (MPI + OpenMP) для параллельного обучения модели линейной регрессии методом стохастического градиентного спуска (SGD) на крупном наборе данных. Опишите:

Как данные будут распределены между узлами кластера (MPI) и внутри узлов (OpenMP).

Каким образом будет организована синхронизация градиентов.

Как будет реализована балансировка нагрузки.

Напишите псевдокод ключевых частей программы.

Указания к выполнению: Используйте паттерн Data Parallelism. MPI-процессы могут работать с разными частями (батчами) данных. OpenMP-потоки внутри процесса могут параллельно вычислять градиенты на своем батче. Для синхронизации градиентов используйте MPI_Allreduce.

Задание 5. Профилирование и поиск узких мест

Для предоставленного преподавателем параллельного приложения (содержащего OpenMP и/или MPI код) выполните:

Сбор метрик производительности с помощью профилировщика (например, perf, gprof для OpenMP; mpiP или IPM для MPI).

Анализ полученного профиля: определите функции с наибольшим временем выполнения, выявите узкие места (недостаточный параллелизм, неэффективные коммуникации, конфликты памяти).

Сформулируйте конкретные рекомендации по оптимизации кода.

Указания к выполнению: Представьте отчет, содержащий скриншоты/графики из профилировщика, их интерпретацию и список предложений по улучшению производительности с обоснованием.

4. КОНТРОЛЬНЫЕ ТЕСТЫ (30 ЗАДАНИЙ)

Тест 1.

Вопрос: Какой из перечисленных видов параллелизма является основным для архитектуры GPU (графического процессора)?

Варианты ответов:

A) Параллелизм на уровне процессов (MIMD)

B) Параллелизм на уровне потоков с общей памятью

C) Параллелизм на уровне данных (SIMD/SIMT)

D) Векторный параллелизм на уровне инструкций (ILP)

Верно: C) Параллелизм на уровне данных (SIMD/SIMT)

Тест 2.

Вопрос: Закон Амдала устанавливает предельное ускорение параллельной программы в зависимости от...

Варианты ответов:

A) Частоты процессора

B) Доли последовательного кода в программе

C) Объема оперативной памяти

D) Скорости сети

Верно: B) Доли последовательного кода в программе

Тест 3.

Вопрос: Какая директива OpenMP создает новую параллельную область, в которой группа потоков совместно выполняет заключенный в структурный блок код?

Варианты ответов:

A) `#pragma omp for` (используется для распараллеливания циклов внутри существующей параллельной области)

B) `#pragma omp parallel` **(создает параллельную область, порождая группу потоков)**

C) `#pragma omp single` (указывает, что блок кода должен быть выполнен только одним потоком в группе)

D) `#pragma omp critical` (определяет критическую секцию для синхронизации)

Верный ответ: B) `#pragma omp parallel`

Тест 4.

Вопрос: Для чего используется клауза private(var) в директиве OpenMP?

Варианты ответов:

A) Для создания отдельной копии переменной var **для каждого потока**

B) Для объявления общей переменной для всех потоков

C) Для синхронизации доступа к переменной var

D) Для уменьшения времени жизни переменной var

Верно: A) Для создания отдельной копии переменной var **для каждого потока**

Тест 5.

Вопрос: Что такое «гонка данных» (race condition)?

Варианты ответов:

A) Состояние, когда программа выполняется быстрее, чем ожидалось

B) Неопределенное поведение программы, возникающее, когда несколько потоков обращаются к одним и тем же данным, и хотя бы один из потоков изменяет их

C) Ошибка компиляции параллельного кода

D) Метод синхронизации с помощью атомарных операций

Верно: B) Неопределенное поведение программы, возникающее, когда несколько потоков обращаются к одним и тем же данным, и хотя бы один из потоков изменяет их

Тест 6.

Вопрос: Первой функцией, которую должен вызвать каждый процесс в MPI-программе, является...

Варианты ответов:

A) `MPI_Finalize`

B) `MPI_Init`

C) `MPI_Comm_rank`

D) `MPI_Send`

Верно: B) `MPI_Init`

Тест 7.

Вопрос: Какой коммуникатор в MPI включает все процессы, участвующие в запущенном параллельном задании?

Варианты ответов:

A) `MPI_COMM_SELF`

B) `MPI_COMM_WORLD`

C) `MPI_COMM_PARENT`

D) `MPI_COMM_NULL`

Верно: B) `MPI_COMM_WORLD`

Тест 8.

Вопрос: Какая коллективная операция MPI используется для сбора данных от всех процессов в один массив на указанном процессе-получателе?

Варианты ответов:

A) `MPI_Bcast`

B) `MPI_Scatter`

C) `MPI_Gather`

D) `MPI_Reduce`

Верно: C) `MPI_Gather`

Тест 9.

Вопрос: Функция MPI_Send в стандартном режиме является...

Варианты ответов:

A) Блокирующей

B) Неблокирующей

C) Синхронной

D) Буферизованной

Верно: A) Блокирующей

Тест 10.

Вопрос: Для безопасного попарного обмена данными между процессами (каждый процесс отправляет и получает) рекомендуется использовать...

Варианты ответов:

A) MPI_Send и MPI_Recv в произвольном порядке

B) MPI_Sendrecv

C) MPI_Isend и MPI_Irecv

D) MPI_Barrier

Верно: B) MPI_Sendrecv

Тест 11.

Вопрос: Каким спецификатором в CUDA объявляется функция, являющаяся ядром, которое вызывается с хоста и выполняется на устройстве?

Варианты ответов:

A) __host__

B) __global__

C) __device__

D) __shared__

Верно: B) __global__

Тест 12.

Вопрос: В иерархии CUDA потоки (threads), которые могут эффективно взаимодействовать через быструю разделяемую память, находятся в пределах одного...

Варианты ответов:

A) Grid (грида)

B) Block (блока)

C) Warp (варпа)

D) Multiprocessor (мультипроцессора)

Верно: B) Block (блока)

Тест 13.

Вопрос: Какая функция CUDA API используется для выделения памяти на устройстве (GPU)?

Варианты ответов:

A) `malloc`

B) `cudaMalloc`

C) `cudaMemcpy`

D) `cudaFree`

Верно: B) `cudaMalloc`

Тест 14.

Вопрос: Для задач какого типа программирование на GPU (CUDA) дает наибольший выигрыш в производительности?

Варианты ответов:

A) Для задач с большим количеством условных переходов и ветвлений

B) Для задач, интенсивно работающих с вводом-выводом

C) Для задач с высокой степенью параллелизма данных и регулярной структурой вычислений

D) Для задач с преимущественно последовательной логикой

Верно: C) Для задач с высокой степенью параллелизма данных и регулярной структурой вычислений

Тест 15.

Вопрос: Что такое warp в архитектуре NVIDIA GPU?

Варианты ответов:

A) Группа из 32 потоков, которые выполняются на мультипроцессоре GPU по принципу SIMT (одна инструкция – множество потоков)

B) Тип памяти с высокой пропускной способностью

C) Инструмент для отладки ядер

D) Способ синхронизации блоков

Верно: A) Группа из 32 потоков, которые выполняются на мультипроцессоре GPU по принципу SIMT (одна инструкция – множество потоков)

Тест 16.

Вопрос: Какой подход к распараллеливанию обучения нейронной сети предпо-

лагает разделение мини-батча данных между несколькими устройствами?

Варианты ответов:

A) Data Parallelism (Параллелизм по данным)

B) Model Parallelism (Параллелизм по модели)

C) Pipeline Parallelism (Конвейерный параллелизм)

D) Task Parallelism (Параллелизм по задачам)

Верно: A) Data Parallelism (Параллелизм по данным)

Тест 17.

Вопрос: Какая коллективная операция MPI наиболее подходит для синхронизации и суммирования градиентов со всех процессов при Data Parallel обучении?

Варианты ответов:

A) MPI_Bcast

B) MPI_Gather

C) MPI_Allreduce

D) MPI_Scatter

Верно: C) MPI_Allreduce

Тест 18.

Вопрос: Распараллеливание какой базовой линейно-алгебраической операции является ключевым для ускорения вычислений в глубоком обучении на GPU?

Варианты ответов:

A) Вычисление определителя матрицы

B) Умножение матриц (GEMM)

C) Обращение матрицы

D) Решение системы линейных уравнений

Верно: B) Умножение матриц (GEMM)

Тест 19.

Вопрос: Какая проблема возникает, когда несколько потоков на разных ядрах модифицируют разные переменные, расположенные в одной кэш-линии?

Варианты ответов:

A) Утечка памяти (memory leak)

B) Ложное разделение кэша (false sharing)

C) Взаимная блокировка (deadlock)

D) Состояние гонки (race condition)

Верно: B) Ложное разделение кэша (false sharing)

Тест 20.

Вопрос: Какой из перечисленных инструментов НЕ является профилировщиком для параллельных программ?

Варианты ответов:

A) Intel VTune Profiler

B) NVIDIA Nsight Systems

C) Scalasca

D) Apache Kafka

Верно: D) Apache Kafka

Тест 21.

Вопрос: Директива `#pragma omp single` в OpenMP указывает, что...

Варианты ответов:

A) Следующий за ней цикл должен быть выполнен одним потоком

B) Заключенный в блок код должен быть выполнен одним (любым) потоком группы, остальные потоки ждут в неявной барьерной точке

C) Следующая переменная должна быть приватной для одного потока

D) Создается одна параллельная задача

Верно: B) Заключенный в блок код должен быть выполнен одним (любым) потоком группы, остальные потоки ждут в неявной барьерной точке

Тест 22.

Вопрос: Для чего используется клауза `reduction(operation:variable)` в OpenMP?

Варианты ответов:

A) Для выполнения указанной операции (например, сложения) над приватными копиями переменной из всех потоков с записью результата в общую переменную

B) Для уменьшения количества потоков в параллельной области

C) Для сокращения объема используемой памяти

D) Для оптимизации доступа к переменной в кэше

Верно: A) Для выполнения указанной операции (например, сложения) над

приватными копиями переменной из всех потоков с записью результата в общую переменную

Тест 23.

Вопрос: Как в CUDA организовать объединенный (coalesced) доступ к глобальной памяти для оптимальной производительности?

Варианты ответов:

A) Каждый поток должен обращаться к произвольным адресам

B) Потоки внутри варпа должны обращаться к последовательным адресам памяти, образуя непрерывный сегмент

C) Использовать только константную память

D) Выделять память на хосте и копировать небольшими порциями

Верно: B) Потоки внутри варпа должны обращаться к последовательным адресам памяти, образуя непрерывный сегмент

Тест 24.

Вопрос: Какой принцип помогает избежать взаимных блокировок (deadlock) при использовании нескольких мьютексов?

Варианты ответов:

A) Всегда захватывать мьютексы в произвольном порядке

B) Установить и соблюдать строгий глобальный порядок захвата мьютексов всеми потоками

C) Никогда не использовать более одного мьютекса

D) Освобождать мьютексы в порядке, обратном захвату

Верно: B) Установить и соблюдать строгий глобальный порядок захвата мьютексов всеми потоками

Тест 25.

Вопрос: Что измеряет метрика «сильное масштабирование» (strong scaling)?

Варианты ответов:

A) Изменение времени решения фиксированной задачи при увеличении числа процессоров

B) Способность решать все более крупные задачи при увеличении числа процессоров

C) Эффективность использования памяти

D) Зависимость производительности от объема данных

Верно: A) Изменение времени решения фиксированной задачи при увеличении числа процессоров

(Тесты 26-30 являются вариациями или углублением предыдущих тем)

Тест 26.

Вопрос: Для чего используется директива `#pragma omp task`?

Верно: Для выражения асинхронного, нерегулярного или рекурсивного параллелизма, где единицей работы является задача, а не итерация цикла.

Тест 27.

Вопрос: Какая функция MPI используется для создания новых коммуникаторов путем разделения существующего?

Верно: `MPI_Comm_split`.

Тест 28.

Вопрос: Какой тип памяти в CUDA характеризуется самым большим объемом, но и самой высокой задержкой?

Верно: Глобальная память (global memory).

Тест 29.

Вопрос: В чем заключается основная идея паттерна "MapReduce"?

Верно: В разделении обработки данных на две фазы: "Map" (независимое преобразование элементов) и "Reduce" (агрегация результатов).

Тест 30.

Вопрос: Какая команда SLURM используется для отправки задания на выполнение в кластер?

Верно: `sbatch` (для пакетного задания) или `srun` (для интерактивного запуска).

5. ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ

Оценка по дисциплине складывается из текущего рейтинга и рейтинга на зачете.

Текущий рейтинг формируется по результатам защиты лабораторных работ (8 работ) и выполнения контрольных тестов (30 тестовых заданий).

За каждую лабораторную работу: **максимум 5 баллов** (корректность реализации – 2, достигнутое ускорение/эффективность – 2, качество отчета/защиты – 1).

Контрольный тест (30 вопросов): **максимум 30 баллов** (1 балл за верный ответ).

Максимальный текущий рейтинг: 70 баллов (85 + 30).

Для допуска к зачету необходимо набрать не менее **28 баллов** в текущем рейтинге и сдать все лабораторные работы.

Рейтинг на зачете определяется следующим образом:

Ответ на теоретический вопрос по билету – **до 10 баллов**.

Решение практического задания (анализ кода, проектирование) – **до 10 баллов**.

Ответы на дополнительные вопросы в рамках курса – **до 10 баллов**.

Максимальный рейтинг на зачете: 30 баллов.

Минимальный проходной балл на зачете – **18 баллов**.

Критерии оценивания на зачете:

Ответ на теоретический вопрос (9-10 баллов): студент демонстрирует полное понимание вопроса, точное знание определений, принципов, моделей. Ответ логически выстроен, приведены примеры.

(7-8 баллов): ответ в целом правильный, но допущены незначительные неточности или неполное раскрытие некоторых аспектов.

(5-6 баллов): студент понимает суть вопроса, но изложение фрагментарно, допущены ошибки в деталях.

Менее 5 баллов: существенные пробелы в знаниях, непонимание основных принципов.

Решение практического задания (9-10 баллов): предложенное решение корректно, эффективно, хорошо обосновано. Студент демонстрирует умение применять теорию на практике.

(7-8 баллов): решение в основном корректно, но содержит неоптимальные элементы или неполное обоснование.

(5-6 баллов): решение имеет существенные недостатки (ошибки в выборе технологии, неучет важных ограничений), но общее направление верное.

Менее 5 баллов: решение неверно или отсутствует.

Итоговый расчет:

Суммарный максимальный рейтинг: 100 баллов (70 текущих + 30 за зачет).

Оценка «Зачтено» выставляется, если студент набрал **в сумме не менее 60 баллов**, при условии получения не менее 28 баллов в текущем рейтинге и не менее 18 баллов на зачете.

Оценка «Не зачтено» выставляется, если суммарный балл ниже 60, или не выполнены минимальные пороги по текущему рейтингу или зачету.

СВЕДЕНИЯ О ДОПОЛНЕНИЯХ И ИЗМЕНЕНИЯХ, ВНЕСЕННЫХ В ФОС ДИСЦИПЛИНЫ

Учебный год	Решение кафедры (№ протокола, дата)	Внесенные в ФОС дополнения и изменения	Подпись заведующего кафедрой