Министерство образования и науки Российской Федерации
Технический институт (филиал) федерального государственного автономного образовательного учреждения высшего образования «Северо-Восточный федеральный университет имени М.К. Аммосова» в г. Нерюнгри
КАФЕДРА МАТЕМАТИКИ И ИНФОРМАТИКИ

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

Б1.В.ДВ.04.01 Параллельное программирование

для программы бакалавриата
по направлению подготовки
01.03.02. "Прикладная математика и информатика",
профиль «Системное программирование и компьютерные технологии»
Форма обучения: очная

Нерюнгри, 2021

УТВЕРЖДЕНО на заседании
выпускающей кафедры МиИ
«14» 05 2021 г., протокол № 10
Заведующий кафедрой _____ / Самохина В.М.
«14» 05 2021 г.

УТВЕРЖДЕНО на заседании
обеспечивающей кафедры МиИ
«14» 05 2021 г., протокол № 10
Заведующий кафедрой _____ / Самохина В.М.
«14» 05 2021 г.

СОГЛАСОВАНО:
Эксперты[1]:

Ф.И.О., должность, организация _____ подпись

Ф.И.О., должность, организация _____ подпись

СОСТАВИТЕЛЬ (И):

Похорукова М.Ю., доцент кафедры МиИ, ТИ (ф) СВФУ _____
Ф.И.О., должность, организация подпись

[1] Эксперт первый: со стороны выпускающей кафедры (или работодатель). Эксперт второй: со стороны обеспечивающей кафедры.

Паспорт фонда оценочных средств
Б1.В.ДВ.04.01 Параллельное программирование

№	Контролируемые разделы (темы)	Код контролируемой компетенции (или ее части)	Требования к уровню усвоения компетенции	Наименование оценочного средства
1.	Введение в параллельное программирование	ОПК-3: способностью к разработке алгоритмических и программных решений в области системного и прикладного программирования , математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям. ПК-4: способностью работать в составе научно-исследовательского и производственного коллектива и решать задачи профессиональной деятельности. ПК-7: способностью к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения	**Знать:** принципы построения архитектуры программного обеспечения и виды архитектуры программного обеспечения; типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке программного обеспечения; методы и средства проектирования программного обеспечения; методы и средства проектирования баз данных; методы и средства проектирования программных интерфейсов; методы и приемы формализации и алгоритмизации поставленных задач; алгоритмы решения типовых задач, области и способы их применения; языки программирования, стандартные библиотеки языков программирования; методологии разработки программного обеспечения; особенности выбранной среды программирования и системы управления базами данных; методы и приемы отладки программного кода; типы и форматы сообщений об ошибках. **Уметь:** использовать существующие типовые решения и шаблоны проектирования программного обеспечения; применять методы и средства проектирования программного обеспечения, структур данных, баз данных, программных интерфейсов; осуществлять коммуникации с заинтересованными сторонами; применять стандартные алгоритмы решения задач в соответствующих областях; применять выбранные языки программирования и среды программирования, системы управления базами данных при разработке программного обеспечения; выявлять ошибки в программном коде, использовать современные компиляторы и отладчики программного кода. **Владеть:** навыками разработки, изменения и согласования архитектуры программного обеспечения с системным аналитиком и архитектором программного обеспечения; навыками проектирования структур данных, баз данных, программных интерфейсов; навыками оценки и согласования сроков выполнения поставленных задач; навыками формализованного описания решений; навыками разработки алгоритмов решения поставленных задач в соответствии с требованиями технического задания; навыками оценки и согласования сроков выполнения поставленных задач; навыками создания программного кода в соответствии с техническим заданием и с использованием специализированных программных средств; навыками анализа и проверки программного кода, его отладки на уровне программных модулей и межмодульных взаимодействий.	лабораторные занятия, самостоятельные работы, аттестационная работа
2.	Модели распределенного программирования. Уровни параллелизма			
3.	Стандарты параллелизма			

Лабораторные работы

В период освоения дисциплины студенты посещают лекционные занятия, самостоятельно изучают дополнительный теоретический материал к лабораторным занятиям. Критериями оценки работы на лабораторных занятиях является: полнота и правильность выполненного задания; степень осознанности, понимания изученного; оформление задания.

Содержание отчета.

1. Титульный лист: название дисциплины; номер и наименование работы; фамилия, имя, отчество студента; дата выполнения.
2. Задание.
3. Листинг программы с выполнением задания.
4. Результаты работы программы.
5. Вывод по работе, сформулированные из цели.

Темы лабораторных работ

Тема 1. Введение в параллельное программирование.

Задания к лабораторной работе

1. Разработать две программы. Первая вычисляет сумму и произведение чисел от L до U, где L – это нижняя граница диапазона, U – верхняя граница диапазона, границы вводятся пользователем, и выводит полученные значения на экран. Вторая программа запускает первую в качестве вновь созданного процесса.
2. Разработать две программы. Первая вычисляет число Фибоначчи по номеру, введенному пользователю, и формуле Fi = Fi–1 + Fi–2, F0 = F1= 1 и
3. выводит его на экран. Вторая программа запускает первую в качестве вновь созданного процесса.
4. Разработать две программы. Первая принимает от пользователя строку, хранящую знаковое целое число, и выводит на экран строковый эквивалент этого числа прописью (например, ввод «-1211» должен приводить к выводу «минус тысяча двести одиннадцать»). Вторая программа запускает первую в качестве вновь созданного процесса.
5. Разработать две программы. Первая принимает от пользователя строку, хранящую число со знаком и плавающей точкой, и выводит на экран строко-вый эквивалент этого числа прописью (например, ввод «-12.11» должен при-водить к выводу «минус двенадцать целых одиннадцать сотых»). Вторая программа запускает первую в качестве вновь созданного процесса.
6. Разработать две программы. Первая принимает от пользователя две строки. Далее, если обе строки хранят целые числа со знаком, то на экран выводится сумма чисел, в противном случае – конкатенация двух введенных строк. Вторая программа запускает первую в качестве вновь созданного
7. процесса.
8. Разработать две программы. Первая принимает от пользователя две прямоугольных матрицы, а затем выводит на экран их сумму и произведение. Вторая программа запускает первую в качестве вновь созданного процесса.
9. Разработать две программы. Первая принимает от пользователя одномерный целочисленный массив, упорядочивает его по возрастанию любым из так называемых «улучшенных алгоритмов» сортировки массивов и выводит на экран. Вторая программа запускает первую в качестве вновь созданного процесса.
10. Разработать две программы. Первая принимает от пользователя одномерный массив чисел с плавающей точкой, упорядочивает его по убыванию любым из так называемых «улучшенных алгоритмов» сортировки массивов и выводит на экран. Вторая программа запускает первую в качестве вновь созданного процесса.
11. Разработать две программы. Первая принимает от пользователя одномерный массив строк, упорядочивает его любым из так называемых «улучшенных алгоритмов» сортировки массивов и выводит на экран. Вторая программа запускает первую в качестве

вновь созданного процесса.

12. Разработать две программы. Первая принимает от пользователя квадратную матрицу, вычисляет сумму элементов, лежащих на главной и побочной диагоналях, и выводит на экран. Вторая программа запускает первую в качестве вновь созданного процесса.

13. Разработать программу, которая вычисляет сумму и произведение чисел от L до U, где L – это нижняя граница диапазона, U – верхняя граница диапазона. Вычисление суммы и произведения оформить как две функции потока. Значения границ диапазон вводятся пользователем, затем запускаются два требуемых потока, а потом на экран выводится полученные значения.

14. Разработать программу, которая вычисляет число Фибоначчи по номе-ру, введенному пользователю, и формуле Fi = Fi–1 + Fi–2, F0 = F1 = 1. Вычисление числа Фибоначчи оформить как функцию потока. По завершению функции потока программа выводит число на экран.

15. Разработать программу для перевода целого числа со знаком в его строковый эквивалент прописью. Перевод числа оформить как функцию по-тока. Ввод числа происходит до запуска потока, а вывод строки – по его завершению. Например, ввод «-1211» должен приводить к выводу «минус тысяча двести одиннадцать».

16. Разработать программу для перевода знакового числа с плавающей точкой в его строковый эквивалент прописью. Перевод числа оформить как функцию потока. Ввод числа происходит до запуска потока, а вывод строки – по его завершению. Например, ввод «-12.11» должен приводить к выводу «минус двенадцать целых одиннадцать сотых».

17. Разработать программу, осуществляющую ввод двух строк, введенных пользователем. Далее, если обе строки хранят целые числа со знаком, то на экран выводится сумма чисел, в противном случае – конкатенация двух введенных строк. Проверку на соответствие строки целому числу, вычисление суммы чисел и конкатенации строк оформить как три разных функции потока. Ввод строк осуществляется до запуска всех потоков, а вывод результатов – после их завершения.

18. Разработать программу, вычисляющую сумму и произведение двух мат-риц. Выполнение этих операций оформить как две функции потока.

19. Сначала программа осуществляет ввод элементов матриц, далее запускает оба потока, а затем выводит результаты на экран.

20. Разработать программу для упорядочивания одномерного целочислен-ного массива. Сортировка массива по возрастанию должна осуществляться любым из так называемых «улучшенных алгоритмов» сортировки и оформляется как функция потока. Сначала выполняется ввод элементов матрицы, затем запускается поток и далее – вывод упорядоченного массива.

21. Разработать программу для упорядочивания одномерного массива чи-сел с плавающей точкой. Сортировка массива по возрастанию должна осу-ществляться любым из так называемых «улучшенных алгоритмов» сортиров-ки и оформляется как функция потока. Сначала выполняется ввод элементов матрицы, затем запускается поток и далее – вывод упорядоченного массива.

22. Разработать программу для упорядочивания одномерного массива строк. Сортировка массива по возрастанию должна осуществляться любым из так называемых «улучшенных алгоритмов» сортировки и оформляется как функция потока. Сначала выполняется ввод элементов матрицы, затем запус-кается поток и далее – вывод упорядоченного массива.

23. Разработать программу для вычисления суммы элементов, лежащих на главной и побочной диагоналях квадратной матрицы. Выполнение этой операции оформляется как функция потока. Ввод элементов матрицы осуществляется до запуска потока, а вывод полученного значения – по его завершению.

Тема 2. Модели распределенного программирования. Уровни параллелизма.

1. Умножение разреженных матриц. Матрица хранится ненулевыми элементами строки. Сравнить производительность программы для матриц разного размера и при разном количестве рабочих процессов.
2. Пусть дана разреженная матрица, представленная ненулевыми элементами строки. Найти транспонированную матрицу. Сравнить производительность программы для матриц разного размера и при разном количестве рабочих процессов.
3. Задача аппроксимации интеграла непрерывной функции f (x) = sinxex на интервале [a,b]. Интервал разбить на подынтервалы по количеству процессоров, для каждого использовать алгоритм адаптивной квадратуры. Алгоритм адаптивной квадратуры:

1.Вычислить m — середину отрезка [a,b].

2.По правилу трапеций или по правилу Симпсона найти аппроксимацию интеграла на отрезках [a,m], [m,b] и [a,b].

3.Если сумма меньших площадей равна большей с некоторой заданной точностью, то аппроксимацию считаем достаточной.

4.Если нет, то процесс повторяется для отрезков [a,m] и [m,b].

4. Задача о 8 ферзях. Нужно расставить на шахматной доске 8 ферзей так, чтобы они не атаковали друг друга. Найти все 92 решения (или требуемое количество), используйте модель «управляющий-рабочие». Управляющий задает начальные позиции ферзей. Рабочий процесс ищет все решения, результат либо передает управляющему, либо выводит в файл.
5. Задача подсчѐта количества уникальных слов в словаре (ни одна буква не повторяется). Использовать модель «управляющий-рабочие» или «взаимодействующие равные». Кроме этого вывести самые длинные слова.
6. Задача сглаживания изображения. Представить изображение матрицей чисел, определяющих цвет пикселя (1 —есть цвет, 0 —нет). Нужно убрать «пики» и «зазубрины», затемнить все светлые пиксели, у которых как мини-мум d соседей не освещены, то же сделать и для тѐмных пикселей. Использовать алгоритм пульсации (см. задание 6). Рассмотреть задачу для разных изображений, разного количества процессоров, числа d.
7. Задача «эволюция». Дано двумерное поле клеток, каждая из которых либо содержит организм (1), либо пуста (0). Каждая клетка проверяет со-стояние своих соседей (их 8) и изменяет своѐ по правилам:

1.Живая клетка, вокруг которой < 2 живых клеток, умирает от одиночества.

2.Живая клетка, вокруг которой есть 2 или 3 живых клеток, выживает.

3.Живая клетка, вокруг которой > 3 живых клеток, умирает от перенасе-ления.

4.Пустая клетка, рядом с которой равно 3 живых соседа, оживает.

С помощью алгоритмы пульсации (см. задание 6) показать n шагов эволюции жизни. Константы «2» и «3» можно заменить своими значениями.

8. Нахождение простых чисел < n решетом Эратосфена. Использовать конвейер процессов-фильтров, каждый получает поток чисел от соседа слева и отправляет поток соседу справа (последний — первому). Первое число, по-лученное фильтром, будет простым; отсылаем соседу числа, не кратные пер-вому. Конец списка можно отслеживать специальным маркером, тогда про-цесс завершает работу.
9. Генерация простых чисел < n с помощью портфеля задач. Используется алгоритм «управляющий-рабочие». Управляющий передаѐт рабочим процес-сам числа, кандидаты в простые (например, 5, 7, 9, ...). Рабочие процессы имеют начальный массив простых чисел (2, 3), получает кандидата на простоту, деля его на меньшие простые числа, полученное простые числа пере-даются управляющему, который и может рассылать их остальным процесам.
10. Решение СЛАУ методом исключений Гаусса. Сравнитьпроизводи-тельность программы для систем разного размера и при разном количестве рабочих процессов.Задание 14Выполнить LU-разложение матрицы . Проверить корректность вычислений, т. е., что LU =

A. Здесь L — нижняя треугольная матрица, U —верхняя треугольная матрица. Сравнить производительность программы для матриц разного размера и при разном количестве рабочих процессов.

11. Найти обратную матрицу. Проверить корректность вычислений. Срав-нить производительность программы для матриц разного размера и при раз-ном количестве рабочих процессов.

12. Промоделировать движение шаров на бильярдном столе. Задать начальные положения и скорости всех шаров, считая движение идеальным, без трения.

13. Найти кратчайшие пути между всеми парами вершин непустого взвешенного графа, если в нѐм нет циклов отрицательной суммарной длины, либо обнаружить наличие таких циклов. Для решения задачи использовать параллельную реализацию алгоритма Флойда.Задание 21Для взвешенного неориентированного графа найти его поддерево, со-единяющее все вершины и обладающее минимальным суммарным весом рѐбер (минимальное остовное дерево). Для решения задачи использовать параллельную реализацию алгоритма Прима.

14. Найти разбиение множества вершин заданного неориентированного графа на непересекающиеся подмножества с максимально близким количеством вершин в каждом из подмножеств и минимальным количеством рѐбер, проходящих между полученными подмножествами. Для решения задачи использовать параллельную реализацию алгоритма Кернигана-Лина.

Тема 3. Стандарты параллелизма

Задание № 1

Написать программу по перемножению матриц по стандартному алгоритму и по алгоритму Винограда. Распараллелить её, используя MPI. Сравнить быстродействие по сравнению с последовательным вариантом и в зависимости от числа используемых процессов.

Задание № 2

Исследование эффективности реализации двухсторонних передач данных.

Постановка задачи. Разработать алгоритм передачи данных между процессами по кругу:

–второй процесс передаёт данные первому процессу;

–третий процесс, получив данные от нулевого, передаёт их второму,

–i-й процесс принимает данные от (i-1) процесса и пересылает их (i+1) процессу, последний процесс передаёт данные нулевому и т.д.

Требуется:

–реализовать алгоритм с использованием блокирующих и неблокирующих передач программы;

–исследовать время решения задачи для длин сообщений: 128, 512, 1024, 2048 и 4096 байт;

–число циклов круговой передачи задавать параметром командной строки;

–построить графики времени выполнения одного цикла передачи, усреднённого по числу кругов, для сообщений разной длины и разного числа процессов.

Задание № 3

Распараллелить программу, вычисляющую определённый интеграл, с помощью MPI. В качестве примера взять любой вычисляемый в конечном виде (для проверки результата работы программы) определённый интеграл.

Критерии оценки:

0 баллов - ставится, если студент не выполнил лабораторную работу.

1 балл - ставится, если студент обнаруживает знание и понимание основных положений лабораторной работы, но при выполнении заданий допущены ошибки или задание

выполнено на 50%; оформление работы выполнено недостаточно последовательно (отсутствуют цель/листинг/результаты/выводы).

2 балла - ставится, если студентом при выполнении заданий допущены неточности или задание выполнено на 70%; оформление работы выполнено с ошибками (отсутствуют цель/выводы).

3 балла - ставится, если студент полностью выполнил задание, правильно ответил на теоретические вопросы преподавателя, оформление работы выполнено последовательно и полно (присутствуют цели работы, задания, листинг программ, результаты и выводы).

Самостоятельная работа студента

Включает проработку конспектов лекций, обязательной и дополнительной учебной литературы в соответствии с планом занятия; выполнение заданий. Основной формой проверки СРС является устный фронтальный опрос на занятии или письменные ответы на вопросы для проверки знаний по теме. Самостоятельная работа студента может быть выполнена печатным способом на листах формата А4. Работа должна быть защищена, то есть студент должен отвечать на вопросы преподавателя по тематике работы. По собственному желанию студент может подготовить презентацию к самостоятельной работе, в которой кратко дается описание работы с уместным использованием мультимедийных технологий (изображения, видео- или аудио-информация). В качестве информационных источников для подготовки работы студент может использовать рекомендованную литературу по данной дисциплине.

Темы заданий для самостоятельной работы студентов

СРС 1. Особенности параллельного программирования.

СРС 2. Модели распределенного программирования.

СРС 3. История параллельных вычислительных систем OpenMP.

СРС 4. Распараллеливание с использованием директив компилятора.

СРС 5. MPI, принципы распараллеливания с обменом сообщениями и разделяемой памятью.

СРС 6. Уровни параллелизма.

СРС 7. Стандарт CORBA.

СРС 8. Среды для параллельного и распределенного программирования.

Перечень теоретических вопросов:

1. В чём заключается основное отличие процесса от потока/нити?

2. Какие преимущества даёт технология OpenMP?

3. Оправдано ли использование технологии OpenMP (и вообще многопоточного программирования) при написании пользовательских программ для персонального компьютера?

4. Как реализуется применение OpenMP при написании программ на языке Fortran? На языках C/C++?

5. Перечислите наиболее используемые директивы OpenMP.

6. Перечислите наиболее используемые функции из библиотеки OpenMP.

7. Какие переменные окружения влияют на число создаваемых потоков?

8. Поясните смысл идеологии Single Program Multiple Data на конкретном примере (можно воспользоваться одной из вышеизложенных программ). Назовите главные отличия OpenMP и MPI. В каких случаях какую из этих технологий предпочтительнее использовать?

2. Как на практике реализуется парадигма Multiple Instructions Multiple Data (MIMD)?

3. Какие типы данных определены в MPI? Как они соотносятся с типами данных языка Фортран?

4. Кратко опишите суть технологии MPI.

5. Какие разновидности процедуры mpi_send() существуют? В чем их отличие друг от друга?

6. Укажите основные различия между синхронным (блокирующим) и асинхронным способами передачи сообщений.

7. На основании чего в MPI можно однозначно идентифицировать тот или иной процесс?

8. Последовательная часть программы (до директивы mpi_init(ierr)) будет выполняться всеми процессами одновременно или же только одним процессом?

Критерии оценки**:**

0 баллов – самостоятельная работа не выполнена.

1 балл – демонстрирует, лишь поверхностный уровень выполнения работы, в содержании выполнения задания допущены принципиальные ошибки.

2 балла – ставится тогда, когда студент выполнил самостоятельную работу, но дает не точные ответы на заданные вопросы.

3 балла – ставится тогда, когда студент выполнил самостоятельную работу, показан высокий уровень освоения студентом учебного материала, содержание выполнения задания не содержит ошибок.

Аттсетационная работа

Аттестационная работа предполагает выполнение письменной работы с обязательными практическими примерами по одной из тем: изучение теоретического материала по заданной теме, самостоятельное рассмотрение примеров, анализ различных источников, формулирование выводов. Работа выполняется печатным способом на листах формата А4, объемом 10-15 страниц. Работа должна быть защищена, то есть студент должен отвечать на вопросы преподавателя по тематике работы. Студент должен подготовить презентацию к работе, в которой кратко дается описание работы с уместным использованием мультимедийных технологий (изображения, видео- или аудио-информация).

Структура работы

Титульный лист

Оглавление

Введение

Содержание работы, состоящее из 2-3 глав, в достаточной мере описывающее предметную область с примерами.

Заключение

Список используемых источников

Приложения (при необходимости)

Тематика аттестационных работ

1. Основные парадигмы распределенных вычислений.
2. Аппаратные средства, используемые в параллельном программировании.
3. Алгоритмизация параллельных вычислений.
4. Параллельное программирование с использованием традиционных языков программирования.
5. Методы и средства параллельной обработки информации: параллельные вычислительные методы, параллельные вычислительные системы, параллельное программирование.
6. Средства спецификации параллельных процессов; механизмы взаимодействия асинхронных параллельных процессов; синхронизирующие примитивы.
7. Основные модели программирования на системах с распределенной и обшей памятью.
8. Методы и языки параллельного программирования на системах с распределенной и общей памятью.
9. Основные конструкции и приемы программирования на системах с распределенной и обшей памятью: язык MPI, язык OpenMP.
10. Применения языков для решения практических задач; сравнение языков.
11. Эффективность параллельных алгоритмов; сравнительные характеристики программ.
12. Параллельная обработка информации на распределенных системах и системах с общей памятью.

Критерии оценки:

0 баллов – контрольная работа не выполнена.

1-10 баллов – демонстрирует, лишь поверхностный уровень выполнения работы, в содержании выполнения задания допущены принципиальные ошибки, на заданные вопросы отвечает нечетко и неполно.

11-34 баллов – ставится тогда, когда студент выполнил контрольную работу, твердо знает материал, но дает не точные ответы на заданные вопросы, в содержании выполнения задания допущены непринципиальные ошибки.

35-40 баллов – ставится тогда, когда студент выполнил контрольную работу, показан высокий уровень освоения студентом учебного материала, содержание выполнения задания не содержит ошибок или допущены неточности, которые были устранены после замечаний, в работе присутствуют четкие и обоснованные выводы.