МИНОБРНАУКИ РОССИИ
Юго-Западный государственный университет

УТВЕРЖДАЮ:
Заведующий кафедрой
программной инженерии
_____ А.В. Малышев
(подпись, инициалы, фамилия)
«30» августа 2024 г.

ОЦЕНОЧНЫЕ СРЕДСТВА
для текущего контроля успеваемости
и промежуточной аттестации обучающихся
по дисциплине
Кластерные системы
(наименование дисциплины)

09.04.04 Программная инженерия, направленность (профиль)
«Предпринимательство, инновации и технологии будущего в программной инженерии»
(код и наименование ОПОП ВО)

Курск – 2024

1 ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ

1.1 ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ

Раздел (тема) дисциплины

«Введение в дисцмплину»

1. Понятие распределённых систем обработки информации
2. Перечислите основные цели применения распределённых систем.
3. Перечислите условия реализации распределённых систем.
4. Опишите возможные структуры распределённых систем.
5. В чём достоинства и недостатки распределённых систем?
6. Чем распределённая система отличается от других параллельных систем обработки информации?
7. Классификация систем с параллельной обработкой
8. Оценка эффективности работы распределённых систем
9. Обмен данными в распределенных системах
10. Структуры распределённых систем

Раздел (тема) дисциплины

«Программное обеспечение кластерных систем.»

11. Требуемые свойства систем распределенной обработки информации
12. программное обеспечение вычислительных систем
13. Охарактеризуйте основные типы программных продуктов
14. Логические слои прикладного программного обеспечения вычислительных систем
15. Понятие и назначение промежуточного слоя программного обеспечения распределенных вычислений
16. Какую основную задачу решает презентационный слой прикладного программного обеспечения?
17. Дайте определение понятиям «клиент» и «сервер».

18.конфигурации «тонкий» и «толстый» клиент.

19.Поясните принципиальное отличие конфигураций «тонкий» и «толстый» клиент.

20.Каковы их преимущества и недостатки?

Раздел (тема) дисциплины

«Практическая реализация кластерных систем»

21.Варианты архитектурного построения систем распределенной обработки информации

22. Распределенная обработка информации на базе механизма удаленного вызова процедур

23.Объектно-ориентированный подход к организации распределенной обработки информации

24.Реализация распределенной обработки информации на основе транзакционного взаимодействия

25.Распределенная обработка информации на основе технологий обмена сообщениями

26.Технология MPI

27.Распределенная обработка информации на основе моделей согласования

28.Архитектура серверов приложений распределенных систем

29.Концепции Grid-технологии

30.Организация распределенной обработки информации на основе Web-технологий

Критерии оценки:

- продемонстрировано непонимание проблемы, ответы неправильные или отсутствуют – 0 баллов.

- продемонстрировано частичное понимание проблемы, доля правильных ответов менее 50% - 6 баллов

- продемонстрировано значительное или полное понимание проблемы, доля правильных ответов более 50% - 12 балла

2 ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ

2.1 БАНК ВОПРОСОВ И ЗАДАНИЙ В ТЕСТОВОЙ ФОРМЕ

Промежуточная аттестация по дисциплине проводится в форме экзамена, проводится в форме компьютерного тестирования.

Для тестирования используются контрольно-измерительные материалы (КИМ) – задания в тестовой форме, составляющие банк тестовых заданий (БТЗ) по дисциплине, утвержденный в установленном в университете порядке.

Проверяемыми на промежуточной аттестации элементами содержания являются темы дисциплины. Все темы дисциплины отражены в КИМ в равных долях (%). БТЗ включает в себя не менее 100 заданий и постоянно пополняется. Для проверки знаний используются вопросы и задания в различных формах:

1. закрытой (с выбором одного или нескольких правильных ответов),
2. открытой (необходимо вписать правильный ответ),
3. на установление правильной последовательности,
4. на установление соответствия.

Умения, навыки и компетенции проверяются с помощью задач (ситуационных, производственных или кейсового характера) и различного вида конструкторов. Все задачи являются многоходовыми. Некоторые задачи, проверяющие уровень сформированности компетенций, являются многовариантными. Часть умений, навыков и компетенций прямо не отражена в формулировках задач, но они могут быть проявлены обучающимися при их решении.

В каждый вариант КИМ включаются задания по каждому проверяемому элементу содержания во всех перечисленных выше формах и разного уровня сложности. Такой формат КИМ позволяет объективно определить качество освоения обучающимися основных элементов содержания дисциплины и уровень сформированности компетенций.

Задания в закрытой форме

1.MPI уменьшает сложность разработки параллельных ...

MPI уменьшает сложность разработки параллельных программ:

- ☐ за счет использования общей памяти
- ☐ большая часть основных операций передачи данных предусматривается стандартом MPI,
- ☐ имеется большое количество библиотек параллельных методов, созданных с использованием MPI.

2.Shedule - способ распределения нагрузки #pragma omp for shedule(...

Shedule - способ распределения нагрузки
#pragma omp for shedule(type[, size]) type=

- ☐ omp
- ☐ parallel
- ☐ runtime
- ☐ static

3.Архитектура суперкомпьютера, в которой каждый ...

Архитектура суперкомпьютера, в которой каждый процессор имеет собственную оперативную память

- ○ симметрмчная
- ○ асимметричная мультипроцессорная обработка
- ○ последовательная

4.Базовые типы данных MPI

Базовые типы данных MPI

- long double
- MPI_unsigned int
- MPI_UNSIGNED_CHAR
- MPI_UNSIGNED

5.Библиотека функций OpenMP

Библиотека функций OpenMP

- void omp_set_threads(int num threads)
- void omp_get_max_threads(void)
- void omp_set_ num_threads(int num threads)
- int omp_get_max_threads(void)

6.Виртуальные топологии в MPI

Виртуальные топологии в MPI

- отсутствует такое понятие
- используется это понятие алгоритмического языка С
- логическая топология линий связи между процессами имеет структуру полного графа (независимо от наличия реальных физических каналов связи между процессорами)
- в MPI имеются средства и для формирования логических (виртуальных) топологий любого требуемого типа

7.Виртуальные топологии в MPI

Виртуальные топологии в MPI

- отсутствует такое понятие
- используется это понятие алгоритмического языка С
- логическая топология линий связи между процессами имеет структуру полного графа (независимо от наличия реальных физических каналов связи между процессорами)
- в MPI имеется возможность представления множества процессов в виде *решетки* произвольной размерности. При этом, граничные процессы решеток могут быть объявлены соседними и, тем самым, на основе решеток могут быть определены структуры типа *тор*

8.Возможные параметры (clause) директивы for [clause ...]

Возможные параметры (clause) директивы for [clause ...]

- sections
- #pragma omp parallel
- lastprivate(list)
- reduction(operator: list)

9.В основу MPI положены следующие основные концепции

В основу MPI положены следующие основные концепции

- когерентность кеша
- понятие парсера
- понятие коммуникатора (группы процессов)
- понятие виртуальной топологии

10.В основу MPI положены следующие основные концепции

В основу MPI положены следующие основные концепции

- когерентность кеша
- понятие парсера
- тип операции передачи сообщения
- тип данных, пересылаемых в сообщении

11.Вычислительное ядро это

Вычислительное ядро это

- совокупность программно-аппаратных средств, способных выполнять поток команд
- программа на языке СИ
- системный блок

12.Группа компьютеров, объединенных высокоскоростными ...

Группа компьютеров, объединенных высокоскоростными каналами связи и представляющая с точки зрения пользователя одну многопроцессорную вычислительную машину

- кластер
- домен
- парсер

13.Действие директивы parallel в технологии OpenMP

Действие директивы parallel в технологии OpenMP

- код одублируется между процессами

- когда основной поток выполнения достигает директиву parallel, создается набор (team) потоков; входной поток является основным потоком этого набора (master thread) и имеет номер 0
- код области дублируется или разделяется между потоками для параллельного выполнения
- в конце области обеспечивается синхронизация потоков - выполняется ожидание завершения потоков; далее все потоки завершаются - дальнейшие вычисления продолжает выполнять только основной поток вычислений всех

14.Директива sections - распределение вычислений для ...

Директива sections - распределение вычислений для раздельных фрагментов кода обеспечивает

- фрагменты выделяются при помощи директивы sections
- распределеляет итерационные блоков
- завершение директивы по умолчанию синхронизируется

15.Директива sections - распределение вычислений для ...

Директива sections - распределение вычислений для раздельных фрагментов кода обеспечивает

- фрагменты выделяются при помощи директивы sections
- распределеляет итерационные блоков
- завершение директивы по умолчанию синхронизируется
- директивы section должны использоваться только в статическом контексте

16.Директива sections - распределение вычислений для ...

Директива sections - распределение вычислений для раздельных фрагментов кода обеспечивает

- фрагменты выделяются при помощи директивы sections
- распределеляет итерационные блоков
- каждый фрагмент выполняется однократно
- разные фрагменты выполняются разными потоками

Директива sections - распределение вычислений для ...

17.Директива sections - распределение вычислений для раздельных фрагментов кода обеспечивает

- фрагменты выделяются при помощи директивы sections
- распределеляет итерационные блоков
- фрагменты выделяются при помощи директивы section
- каждый фрагмент выполняется однократно

18.Директива sections - распределение вычислений для ...

Директива sections - распределение вычислений для раздельных фрагментов кода обеспечивает

- фрагменты выделяются при помощи директивы sections
- распределеляет итерационные блоков
- разные фрагменты выполняются разными потоками

19.Директива sections - распределение вычислений для ...

Директива sections - распределение вычислений для раздельных фрагментов кода обеспечивает

- фрагменты выделяются при помощи директивы sections
- распределеляет итерационные блоков
- каждый фрагмент выполняется однократно

20.Директивы OpenMP Взаимное исключение критических ...

Директивы OpenMP
Взаимное исключение критических интервалов

- в любой момент времени только две нити могут находиться внутри критического интервала
- в любой момент времени только три нити могут находиться внутри критического интервала
- в любой момент времени только одна нить может находиться внутри критического интервала;
- если ни одна нить не находится в критическом интервале, то любая нить, желающая войти в критический интервал, должна получить разрешение без какой либо задержки

21.Директивы OpenMP Взаимное исключение критических ...

Директивы OpenMP
Взаимное исключение критических интервалов

- в любой момент времени только две нити могут находиться внутри критического интервала
- в любой момент времени только три нити могут находиться внутри критического интервала

- ни одна нить не должна бесконечно долго ждать разрешения на вход в критический интервал (если ни одна нить не будет находиться внутри критического интервала бесконечно).
- если ни одна нить не находится в критическом интервале, то любая нить, желающая войти в критический интервал, должна получить разрешение без какой либо задержки

22.Директивы для распределения вычислений в ...

Директивы для распределения вычислений в параллельной области

- #pragma omp
- #pragma omp parallel
- single

23.Директивы для распределения вычислений в ...

Директивы для распределения вычислений в параллельной области

- #pragma omp
- #pragma omp parallel
- sections
- single

24.Для определения ранга процесса используется функция

Для определения ранга процесса используется функция

- int MPI_Comm_size (MPI_Comm comm, int *size)
- int MPI_Comm_rank (MPI_Comm comm, int *rank)
- int MPI_Init (int *agrc, char ***argv)

25.Для передачи данных от всех процессов одному ...

Для передачи данных от всех процессов одному процессу должна быть выполнена функция

- int MPI_Recv(void *buf, int count, MPI_Datatype type, int source,int tag, MPI_Comm comm, MPI_Status *status)
- int MPI_Reduce(void *sendbuf, void *recvbuf,int count, MPI_Datatype type, int root, MPI_Comm comm)
- int MPI_Send(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm)

26.Для передачи данных от одного процесса всем ...

Для передачи данных от одного процесса всем процессам программы должна быть выполнена функция:

- int MPI_Recv(void *buf, int count, MPI_Datatype type, int source,int tag, MPI_Comm comm, MPI_Status *status)
- int MPI_Bcast(void *buf,int count,MPI_Datatype type, int root, MPI_Comm comm)
- int MPI_Send(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm)

27.Для передачи сообщения процесс-отправитель должен ...

Для передачи сообщения процесс-отправитель должен выполнить функцию

- int MPI_Comm_size (MPI_Comm comm, int *size)
- int MPI_Comm_rank (MPI_Comm comm, int *rank)
- int MPI_Send(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm)

28.Для приема сообщения процесс-получатель должен ...

Для приема сообщения процесс-получатель должен выполнить функцию:

- int MPI_Recv(void *buf, int count, MPI_Datatype type, int source,int tag, MPI_Comm comm, MPI_Status *status)
- int MPI_Comm_rank (MPI_Comm comm, int *rank)
- int MPI_Send(void *buf, int count, MPI_Datatype type, int dest, int tag, MPI_Comm comm)

29.Достижение параллелизма возможно только при ...

Достижение параллелизма возможно только при выполнимости следующих требований

- независимость функционирования отдельных устройств ЭВМ
- избыточность элементов вычислительной системы
- наличие нескольких принтеров
- избыточность памяти

30.Единица измерения производительности компьютера

Единица измерения производительности компьютера

- мегафлосп
- бит
- Мгерц
- Кбит

31.Классификации вычислительных систем по "модели ...

Классификации вычислительных систем по "модели программирования"

- ☐ последовательная система
- ☐ симметричные параллельные системы (SMP) - общей памятью
- ☐ массивно-параллельные системы - с распределенной памятью
- ☐ кластерные системы
- ☐ метакомпьютеры
- ☐ параллельная
- ☐ кольцевая

32.Классификации вычислительных систем по назначению

Классификации вычислительных систем по назначению

- ☐ персональный компьютер (рабочая станция)
- ☐ тонкий клиент (X-терминал)
- ☐ сервер
- ☐ мейнфрейм
- ☐ суперкомпьютер
- ☐ смартфон

33.Классификация многопроцессорных вычислительных ...

Классификация многопроцессорных вычислительных систем

- ☐ мультипроцессоры
- ☐ мультикомпьютеры
- ☐ ноутбук
- ☐ сервер

34.классификация по способам взаимодействия ...

классификация по способам взаимодействия последовательностей (потоков) выполняемых команд и обрабатываемых данных:

- SISD
- SIMD
- MISD
- MIMD
- MMSS
- MSMS

35.Кластер это

Кластер это

- множество отдельных компьютеров, объединенных в сеть, для которых при помощи специальных аппаратно-программных средств обеспечивается возможность унифицированного управления (single system image), надежного функционирования (availability) и эффективного использования (performance)
- системный блок
- стример

36.Кластер это

Кластер это

- группа компьютеров, объединенных в локальную вычислительную сеть (ЛВС) и способных работать в качестве единого вычислительного ресурса
- системный блок

- стример
- парсер

37.Конструкции для синхронизации нитей

Конструкции для синхронизации нитей

- директива parallel
- директива section
- семафоры
- директива CRITICAL

38.Конструкции для синхронизации нитей

Конструкции для синхронизации нитей

- директива parallel
- директива section
- директива ATOMIC
- директива MASTER

39.Конструкции для синхронизации нитей

Конструкции для синхронизации нитей

- директива parallel
- директива section
- директива BARRIER
- директива CRITICAL

40.Конструкции для синхронизации нитей

Конструкции для синхронизации нитей

- директива parallel
- директива section
- директива TASKWAIT
- директива FLUSH

41.Место, где физически расположены файлы веб-страниц, ...

Место, где физически расположены файлы веб-страниц, базы данных и пр., а также серверные программы, взаимодействующие с обращающимся к ним браузерам и организующие, по соответствующему запросу браузера, компоновку требуемой веб-страницы.

- Web-сервер
- виртуальный каталог IIS
- SQL-сервер
- База данных

42.Многопроцессорная система - вычислительная система:

Многопроцессорная система - вычислительная система:

- обеспечивающая работу в Интернете
- состоящая из нескольких процессоров
- обеспечивающая параллельную обработку данных

43.Многопроцессорная система - вычислительная система:

Многопроцессорная система - вычислительная система:

- обеспечивающая работу в Интернете

- состоящая из нескольких процессоров
- состоящую из нескольких мониторов
- включающая СУБД

44.Мультикомпьютеры...

Мультикомпьютеры...

- для доступа к данным, располагаемых на других процессорах, необходимо явно выполнить операции передачи сообщений (message passing operations).
- обеспечивается однородный доступ к памяти
- общая оперативная память

45.Мультикомпьютеры...

Мультикомпьютеры...

- не обеспечивают общий доступ ко всей имеющейся в системах памяти (no-remote memory access or NORMA),
- обеспечивается однородный доступ к памяти
- общая оперативная память

46.Мультикомпьютеры...

Мультикомпьютеры...

- каждый процессор системы может использовать только свою локальную память,
- обеспечивается однородный доступ к памяти
- общая оперативная память

47.Мультикомпьютеры...

Мультикомпьютеры...

- не обеспечивают общий доступ ко всей имеющейся в системах памяти (no-remote memory access or NORMA),
- каждый процессор системы может использовать только свою локальную память,
- для доступа к данным, располагаемых на других процессорах, необходимо явно выполнить операции передачи сообщений (message passing operations).
- обеспечивается однородный доступ к памяти
- общая оперативная память

48.Мультикомпьютеры. Данный подход используется при ...

Мультикомпьютеры. Данный подход используется при построении таких типов многопроцессорных вычислительных систем:

- массивно-параллельных систем (massively parallel processor or MPP), например: IBM RS/6000 SP2, Intel PARAGON, ASCI Red, транспьютерные системы Parsytec,
- кластеров (clusters), например: AC3 Velocity и NCSA NT Supercluster
- векторных параллельных процессоров (parallel vector processor or PVP). Примеры: Cray T90,
- симметричных мультипроцессоров (symmetric multiprocessor or SMP). Примеры: IBM eServer, Sun StarFire, HP Superdome, SGI Origin.

49.Мультикомпьютеры это

Мультикомпьютеры это

- системы с общей разделяемой памятью
- системы с распределенной памятью
- группа компьютеров, объединенных в локальную вычислительную сеть (ЛВС) и способных работать в качестве единого вычислительного ресурса
- системный блок

50.Мультипроцессорная обработка ...

Мультипроцессорная обработка ...

- выполнение программы сжания файлов
- работа в браузере
- одновременное выполнение двух и более процессов (программ) несколькими процессорами вычислительной системы

51.Мультипроцессорная обработка ...

Мультипроцессорная обработка ...

- выполнение программы сжания файлов
- работа в браузере
- одновременное выполнение двух и более процессов (программ) несколькими процессорами вычислительной системы
- выполнение программы на кластере

52.Мультипроцессоры с использованием единой общей ...

Мультипроцессоры с использованием единой общей памяти (shared memory)...

- обеспечивается однородный доступ к памяти (uniform memory access or UMA)
- не обеспечивается однородный доступ к памяти
- группа компьютеров, объединенных в локальную вычислительную сеть

53.Мультипроцессоры с использованием единой общей ...

Мультипроцессоры с использованием единой общей памяти (shared memory)...

- обеспечивается однородный доступ к памяти (uniform memory access or UMA)
- не обеспечивается однородный доступ к памяти
- группа компьютеров, объединенных в локальную вычислительную сеть
- являются основой для построения векторных параллельных процессоров
- являются основой для построения симметричных мультипроцессоров

54.Мультипроцессоры с использованием физически ...

Мультипроцессоры с использованием физически распределенной памяти:

- упрощаются проблемы создания мультипроцессоров (известны примеры систем с несколькими тысячами процессоров),
- обеспечивается однородный доступ к памяти
- общая оперативная память

55.Мультипроцессоры с использованием физически ...

Мультипроцессоры с использованием физически распределенной памяти:

- возникают проблемы эффективного использования распределенной памяти (время доступа к локальной и удаленной памяти может различаться на несколько порядков).
- обеспечивается однородный доступ к памяти
- общая оперативная память

56.Мультипроцессоры с использованием физически ...

Мультипроцессоры с использованием физически распределенной памяти:

- упрощаются проблемы создания мультипроцессоров (известны примеры систем с несколькими тысячами процессоров),
- возникают проблемы эффективного использования распределенной памяти (время доступа к локальной и удаленной памяти может различаться на несколько порядков).
- не обеспечивается однородный доступ к памяти
- нет общей оперативной памяти

57.Мультипроцессоры это

Мультипроцессоры это

- системы с общей разделяемой памятью
- группа компьютеров, объединенных в локальную вычислительную сеть (ЛВС) и способных работать в качестве единого вычислительного ресурса
- системный блок

58.Недостатки кластера:

Недостатки кластера:

- мощности отдельных процессоров позволяет строить кластеры из сравнительно небольшого количества отдельных компьютеров
- Организация взаимодействия вычислительных узлов кластера при помощи передачи сообщений обычно приводит к значительным временным задержкам,
- Дополнительные ограничения на тип разрабатываемых параллельных алгоритмов и программ (низкая интенсивность потоков передачи данных)

59.Обобщенная передача данных от всех процессов одному ...

Обобщенная передача данных от всех процессов одному процессу

- int MPI_Recv(void *buf, int count, MPI_Datatype type, int source,int tag, MPI_Comm comm, MPI_Status *status)
- int MPI_Gather(void *sbuf,int scount,MPI_Datatype stype, void *rbuf,int rcount,MPI_Datatype rtype, int root, MPI_Comm comm),
- int MPI_Scatter(void *sbuf,int scount,MPI_Datatype stype, void *rbuf,int rcount,MPI_Datatype rtype, int root, MPI_Comm comm)

60.Обобщенная передача данных от одного процесса всем ...

Обобщенная передача данных от одного процесса всем процессам

- int MPI_Recv(void *buf, int count, MPI_Datatype type, int source,int tag, MPI_Comm comm, MPI_Status *status)
- int MPI_Reduce(void *sendbuf, void *recvbuf,int count, MPI_Datatype type, int root, MPI_Comm comm)

- int MPI_Scatter(void *sbuf,int scount,MPI_Datatype stype, void *rbuf,int rcount,MPI_Datatype rtype, int root, MPI_Comm comm)

61.Объект веб-страницы (текст или изображение), ...

Объект веб-страницы (текст или изображение), устанавливающий связь с другим объектом в сети Интернет

- гиперссылка
- URL
- изображение
- нумерованный список

62.Определение количества процессов в выполняемой ...

Определение количества процессов в выполняемой параллельной программе осуществляется при помощи функции:

- int MPI_Comm_size (MPI_Comm comm, int *size)
- int MPI_Comm_rank (MPI_Comm comm, int *rank)
- int MPI_Init (int *agrc, char ***argv)

63.Основу MPI составляют следующие операции передачи ...

Основу MPI составляют следующие операции передачи сообщений

- между несколькими потоками
- между двумя потоками
- парные (*point-to-point*) операции между двумя процессами
- коллективные (*collective*) коммуникационные действия для одновременного взаимодействия нескольких процессов

64.Параллельный процесс ...

Параллельный процесс ...

- печать результатов выполнения программы на принтере
- работа в Интернете
- совершение процесс, действия которого могут выполняться одновременно

65.Параллельный процесс ...

Параллельный процесс ...

- печать результатов выполнения программы на принтере
- работа в Интернете
- совершение процесс, действия которого могут выполняться одновременно
- одновременное выполнение двух потоков одного процесса

66.Параметр DEFAULT

Параметр DEFAULT

- позволяет создать локальные переменные потоков, которые перед использованием инициализируются значениями исходных переменных
- определяет список переменных, которые будут локальными для каждого потока; переменные создаются в момент формирования потоков параллельной области; начальное значение переменных является неопределенным
- меняет класс переменной по умолчанию

67.Параметр firstprivate

Параметр firstprivate

- позволяет создать локальные переменные потоков, которые перед использованием инициализируются значениями исходных переменных
- определяет список переменных, которые будут локальными для каждого потока; переменные создаются в момент формирования потоков параллельной области; начальное значение переменных является неопределенным
- определяет список переменных, которые будут общими для всех потоков параллельной области; правильность использования таких переменных должна обеспечиваться программистом

68.Параметр lastprivate

Параметр lastprivate

- позволяет создать локальные переменные потоков, которые перед использованием инициализируются значениями исходных переменных
- определяет список переменных, которые будут локальными для каждого потока; переменные создаются в момент формирования потоков параллельной области; начальное значение переменных является неопределенным
- передает значение приватной переменной, посчитанной на последней итерации в глобальную переменную

69.Параметр private

Параметр private

- позволяет создать локальные переменные потоков, которые перед использованием инициализируются значениями исходных переменных

- определяет список переменных, которые будут локальными для каждого потока; переменные создаются в момент формирования потоков параллельной области; начальное значение переменных является неопределенным
- определяет список переменных, которые будут общими для всех потоков параллельной области; правильность использования таких переменных должна обеспечиваться программистом

70.Параметр shared

Параметр shared

- позволяет создать локальные переменные потоков, которые перед использованием инициализируются значениями исходных переменных
- определяет список переменных, которые будут локальными для каждого потока; переменные создаются в момент формирования потоков параллельной области; начальное значение переменных является неопределенным
- определяет список переменных, которые будут общими для всех потоков параллельной области; правильность использования таких переменных должна обеспечиваться программистом

71.Параметр THREADPRIVATE

Параметр THREADPRIVATE

- позволяет создать локальные переменные потоков, которые перед использованием инициализируются значениями исходных переменных
- определяет список переменных, которые будут локальными для каждого потока; переменные создаются в момент формирования

потоков параллельной области; начальное значение переменных является неопределенным

- сохраняют глобальную область видимости внутри каждой нити

71.Первой вызываемой функцией MPI должна быть функция:

Первой вызываемой функцией MPI должна быть функция:

- int MPI_Comm_size (MPI_Comm comm, int *size
- int MPI_Finalize (void)
- int MPI_Init (int *agrc, char ***argv)

72.Передача сообщений в MPI

Передача сообщений в MPI

- для параметра source может быть указано значение MPI_ANY_SOURCE
- отправляемое сообщение определяется через указание блока памяти (*буфера*), в котором это сообщение располагается. Используемая для указания буфера триада (*buf, count, type*) входит в состав параметров практически всех функций передачи данных,
- процессы, между которыми выполняется передача данных, обязательно должны принадлежать коммуникатору, указываемому в функции *MPI_Send*,
- параметр *tag* используется только при необходимости различения передаваемых сообщений, в противном случае в качестве значения параметра может быть использовано произвольное целое число

73.Под параллельной программой в рамках MPI понимается

Под параллельной программой в рамках MPI понимается

- вывод результатов выполнения программы на принтер
- работа в Интернете
- исходный программный код разрабатывается на алгоритмических языках C или Fortran с использованием библиотеки MPIпроцессы могут выполняться на разных процессорах; вместе с этим, на одном процессоре могут располагаться несколько процессов
- количество процессов и число используемых процессоров определяется в момент запуска параллельной программы средствами среды исполнения MPI программ

74.Под параллельной программой в рамках MPI понимается

Под параллельной программой в рамках MPI понимается

- вывод результатов выполнения программы на принтер
- работа в Интернете
- все процессы программы последовательно перенумерованы
- номер процесса именуется рангом процесса

75.Под параллельной программой в рамках MPI понимается

Под параллельной программой в рамках MPI понимается

- вывод результатов выполнения программы на принтер
- работа в Интернете
- множество одновременно выполняемых процессов
- процессы могут выполняться на разных процессорах; вместе с этим, на одном процессоре могут располагаться несколько процессов

76.Положительные стороны технологии OpenMP

Положительные стороны технологии OpenMP

- MPI позволяет существенно снизить сложность разработки параллельных программ за счет использования общей памяти
- MPI позволяет существенно снизить остроту проблемы переносимости параллельных программ между разными компьютерными системами.
- MPI содействует повышению эффективности параллельных вычислений - практически для каждого типа вычислительных систем существуют реализации библиотек MPI.
- MPI уменьшает сложность разработки параллельных программ:

77.Положительные стороны технологии OpenMP

- работа в браузере
- поэтапное (инкрементальное) распараллеливание
- единственность разрабатываемого кода
- эффективность
- переносимость

78.Понятие коммуникаторов...

Понятие коммуникаторов...

- используется для связи между несколькими потоками
- используется для связи между двумя потоками
- в ходе вычислений могут создаваться новые и удаляться существующие коммуникаторы.
- один и тот же процесс может принадлежать разным коммуникаторам

79.Понятие коммуникаторов...

Понятие коммуникаторов...

- ☐ используется для связи между несколькими потоками
- ☐ используется для связи между двумя потоками
- ☐ парные операции передачи данных выполняются для процессов, принадлежащих одному и тому же коммуникатору
- ☐ коллективные операции применяются одновременно для всех процессов коммуникатора

80.Понятие коммуникаторов...

Понятие коммуникаторов...

- ☐ используется для связи между несколькими потоками
- ☐ используется для связи между двумя потоками
- ☐ все имеющиеся в параллельной программе процессы входят в состав создаваемого по умолчанию коммуникатора с идентификатором MPI_COMM_WORLD
- ☐ при необходимости передачи данных между процессами из разных групп необходимо создавать глобальный коммуникатор (*intercommunicator*)

81.Понятие коммуникаторов...

Понятие коммуникаторов...

- ☐ используется для связи между несколькими потоками
- ☐ используется для связи между двумя потоками

- специально создаваемый служебный объект, объединяющий в своем составе группу процессов и ряд дополнительных параметров (*контекст)*
- указание используемого коммуникатора является обязательным для операций передачи данных в MPI

82.Поток это

Поток это

- совокупность программно-аппаратных средств, способных выполнять поток команд
- последовательность инструкций, выполняемвых вычислительным ядром в рамках процесса
- программа на языке СИ
- системный блок

83.Преимущества кластера:

Преимущества кластера:

- могут быть образованы на базе уже существующих у потребителей отдельных компьютеров, либо же сконструированы из типовых компьютерных элементов;
- повышение вычислительной мощности отдельных процессоров позволяет строить кластеры из сравнительно небольшого количества отдельных компьютеров (lowly parallel processing),
- для параллельного выполнения в алгоритмах достаточно выделять только крупные независимые части расчетов (coarse granularity).

- Организация взаимодействия вычислительных узлов кластера при помощи передачи сообщений обычно приводит к значительным временным задержкам,

84.Прием сообщенийв MPI

Укажите правильные суждения

- для параметра tag может быть указано значение MPI_ANY_SOURCE
- Буфер памяти должен быть достаточным для приема сообщения, а тип элементов передаваемого и принимаемого сообщения должны совпадать; при нехватке памяти часть сообщения будет потеряна и в коде завершения функции будет зафиксирована ошибка переполнения,
- При необходимости приема сообщения от любого процесса-отправителя для параметра *source* может быть указано значение *MPI_ANY_SOURCE*,
- При необходимости приема сообщения с любым тегом для параметра *tag* может быть указано значение *MPI_ANY_TAG*

85.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- int i = omp_get_thread_num();
- printf("Hello from thread %d\n", i);
- #pragma omp parallel private(beta,pi)
- #pragma omp parallel default(shared)

86.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- }
- Hello from thread 0
- #pragma omp parallel private(beta,pi)
- #pragma omp parallel default(shared) private(beta,pi)

87.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- &pragma omp parallel private(beta,pi)
- int i = omp_get_thread_num();
- #pragma omp parallel default(shared)
- #pragma omp parallel default(shared) private(beta,pi)

88.ример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- h = 1.0 / (double) n;
- sum = 0.0;
- #pragma omp parallel default (none) private (i,x) shared (n,h,sum)
- #pragma omp parallel default(shared) private(beta,pi)

89.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- int id = omp_get_thread_num();
- int numt = omp_get_num_threads();
- #pragma omp critical

- #pragma omp parallel default (none) private (i,x) shared (n,h,sum)

90.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- pi = h * sum;
- printf("pi is approximately %.16f", pi);
- #pragma omp parallel default(shared)
- #pragma omp critical

91.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- #include
- #include
- #pragma omp parallel private(beta,pi)
- #pragma omp parallel

92.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- #include
- int main()
- #pragma omp parallel private(beta,pi)
- #pragma omp parallel default(shared)

93.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- &pragma omp parallel private(beta,pi)
- int i = omp_get_thread_num();
- #pragma omp parallel default(shared)

94.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- }
- Hello from thread 0
- #pragma omp parallel default(shared) private(beta,pi)

95.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- &pragma omp parallel private(beta,pi)
- "pragma omp parallel private(beta,pi)
- @pragma omp parallel default(shared)
- #pragma omp parallel default(shared) private(beta,pi)

96.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- for (i = id + 1; i <= n; i=i+numt)
- x = h * ((double)i - 0.5);
- #pragma omp parallel default(shared)
- #pragma omp critical

97.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- #include
- int main()
- #pragma omp parallel default(shared)

98.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- int i = omp_get_thread_num();
- printf("Hello from thread %d\n", i);
- #pragma omp parallel private(beta,pi)

99.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- #include
- #include
- #pragma omp parallel private(beta,pi)

100.Пример записи директив в технологии OpenMP

Пример записи директив в технологии OpenMP

- &pragma omp parallel private(beta,pi)
- #pragma omp parallel private(beta,pi)
- #pragma omp parallel default(shared)
- #pragma omp parallel default(shared) private(beta,pi)

101.Принципы организации параллелизма в технологии OpenMP

К принципу относится

- печать результатов выполнения программы на принтере
- работа в Интернете
- при выполнении обычного кода (вне параллельных областей) программа исполняется одним потоком (master thread)
- при появлении директивы #parallel происходит создание "команды" (team) потоков для параллельного выполнения вычислений
- после выхода из области действия директивы #parallel происходит синхронизация, все потоки, кроме master, уничтожаются
- продолжается последовательное выполнение кода (до очередного появления директивы #parallel параллелизм добавляется инкрементально

102.Проблемы с использованием единой общей памяти ...

Проблемы с использованием единой общей памяти мультипроцессора:

- доступ с разных процессоров к общим данным и обеспечение, в этой связи, однозначности (когерентности) содержимого разных кэшей
  (cache coherence problem),
- необходимость синхронизации взаимодействия одновременно выполняемых потоков команд
- не обеспечивается однородный доступ к памяти

103.Распределенная иерархическая база данных, ...

Распределенная иерархическая база данных, содержащая информацию об именах серверов Интернета и позволяющая по имени системы определить ее IP-адрес.

- DNS
- операционная смстема
- парсер
- Domain Name Service

104.режимы передачи данных

режимы передачи данных могут быть следующими

- Параллельный
- Буферизованный (Buffered)
- Режим передачи по готовности (Ready)
- Синхронный (Synchronous)

105.Сервис сети Интернет, позволяющий получать доступ к ...

Сервис сети Интернет, позволяющий получать доступ к массивам информации, размещенным в глобальной сети

- WWW
- http
- TCP/IP
- IP

106.Сетевая операционная система - операционная ...

Сетевая операционная система - операционная система, обеспечивающая

- просмотр фильмов
- файл-серверную технологию
- обработку, хранение и передачу данных в информационной сети

107.Способы реализация параллелизма

Способы реализация параллелизма

- SQL-сервер
- парсер
- SIMD
- MIMD

108.Структура параллельной программы, разработанная с ...

Структура параллельной программы, разработанная с использованием MPI, должна иметь следующий вид:

- #include "mpi.h"
  int main (int argc, char *argv[]) {
  <программный код без использования MPI функций> MPI_Init (&agrc, &argv);
  <программный код с использованием MPI функций > MPI_Finalize();
  <программный код без использования MPI функций > return 0;
  }
- #include "mpi.h"
  int main (int argc, char *argv[]) {
  <программный код без использования MPI функций><программный код с использованием MPI функций > MPI_Finalize();
  <программный код без использования MPI функций > return 0;
  }
- #include "mpi.h"
  int main (int argc, char *argv[]) {
  <программный код без использования MPI функций> MPI (&agrc, &argv);

<программный код с использованием MPI функций > MPI_Finalize();
<программный код без использования MPI функций > return 0;
}

109.Суперкомпьютер это

Суперкомпьютер это

- вычислительная система, обладающая предельными характеристиками по производительности среди имеющихся в каждый конкретный момент времени компьютерных систем
- системный блок
- стример
- парсер

110.Типы данных в MPI

Типы данных в MPI

- отсутствует понятие типов данных
- используется понятие типов данных только из алгоритмического языка С
- при выполнении операций передачи сообщений для указания передаваемых или получаемых данных в функциях MPI необходимо указывать тип пересылаемых данных.
- в MPI имеются возможности для создания новых *производных типов данных* для более точного и краткого описания содержимого пересылаемых сообщений

111.Типы данных в MPI

Типы данных в MPI

- ☐ отсутствует понятие типов данных
- ☐ используется понятие типов данных только из алгоритмического языка С
- ☐ при выполнении операций передачи сообщений для указания передаваемых или получаемых данных в функциях MPI необходимо указывать тип пересылаемых данных.
- ☐ MPI содержит большой набор *базовых типов данных*, во многом совпадающих с типами данных в алгоритмических языках С и Fortran.

Задания в открытой форме

1._____ вычислительные системы состоят из нескольких компьютеров.

2.Закон _____иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей

3._____ вычислительные системы состоят из нескольких процессоров.

4.Закон _____иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей

5.Кластерные системы относятся к _____ вычислительным системам

6.Параллельные вычислительные системы бывают многопроцессорные и_____

7. Параллельные вычислительные системы бывают многомашинные и_____

8.Суперкомпьютеры относятся к _____ вычислительным системам

9._____ команда может служить примером команды, которая позволяет разделять вычислительный поток

10. системы класса МКМД относятся к системам с _____параллелизмом

Задания на установление правильной последовательности:

1.Расположите классы вычислительных систем в порядке возрастания уровней параллелизма:

ОКОД

ОКМД

МКМД

2.Расположите классы вычислительных систем в порядке возрастания уровней параллелизма:

ОКОД

МКОД

МКМД

3. Расположите типы архитектур процессора в порядке возрастания сложности команд

RISC

CISC

Задания на установление соответствия:

1.Установите соответствие:

1. Многопроцессорные системы
2. Многомашинные системы

А) Технология программирования OpenMP

Б) Технология программирования MPI

2. Установите соответствие:

1. Многопроцессорные системы
2. Многомашинные системы

А) Общая оперативная память

Б) Разделенная оперативная память

3.Существуют следующие уровни изоляции транзакций:

Уровень1: Внутри данной транзакции видны только завершённые изменения, сделанные другими транзакциями

Уровень2: Внутри данной транзакции видны все (завершённые и незавершённые) изменения, сделанные другими транзакциями

Уровень3: Внутри данной транзакции видны те данные, которые были в базе на момент начала транзакции

Установите соответствие между значениями свойства TransIsolation компоненты первым уровнем изоляции транзакций.

A) tiDirtyRead

B) tiReadCommited

C) tiRepeatableRead

Шкала оценивания

Тест содержит вопросы из разных категорий. Максимальное количество баллов за решение теста – 30 баллов.

Критерии оценивания результатов тестирования:

Каждый вопрос (задание) в тестовой форме оценивается по дихотомической шкале: выполнено – **2 балла**, не выполнено – **0 баллов**.

2.2 КОМПЕТЕНТНОСТНО-ОРИЕНТИРОВАННЫЕ ЗАДАЧИ

1.Допустим, имеется матрица 4 на 6 вычислительных узлов распределенной вычислительной системы. Реализована коллекторная схема обмена данными. Вычислите, сколько циклов передачи данных нужно выполнить, чтобы организовать полный обмен информацией. Кратко обоснуйте свои ответы.

2.В матрице 6 на 8 вычислительных узлов распределенной вычислительной системы. Реализована циклическая схема обмена данными. Вычислите, сколько циклов передачи данных нужно выполнить, чтобы организовать полный обмен информацией. Кратко обоснуйте свои ответы.

3.По сформулированным Вами параметрам вычислительной системы определите значение ее эффективности

Шкала оценивания решения компетентностно-ориентированной задачи: в соответствии с действующей в университете балльно-рейтинговой системой оценивание результатов промежуточной аттестации обучающихся осуществляется в рамках 100-балльной шкалы, при этом максимальный балл по промежуточной аттестации обучающихся по очной форме обучения составляет 36 баллов, по очно-заочной и заочной формам обучения – 60 (установлено положением П 02.018). Общий балл по промежуточной аттестации суммируется с баллами, полученными обучающимся по результатам текущего контроля успеваемости в течение семестра; сумма баллов переводится в оценку по дихотомической шкале (для зачета) или в оценку по 5-балльной шкале (для экзамена)

Максимальное количество баллов за решение компетентностно-ориентированной задачи – 6 баллов.

Балл, полученный обучающимся за решение компетентностно-ориентированной задачи, суммируется с баллом, выставленным ему по результатам тестирования. Общий балл промежуточной аттестации суммируется с баллами, полученными обучающимся по результатам текущего контроля успеваемости в течение семестра; сумма баллов переводится в оценку следующим образом:

Соответствие 100-балльной и дихотомической шкал

Сумма баллов по 100-балльной шкале	*Оценка по дихотомической шкале*
100–50	зачтено
49 и менее	не зачтено

Соответствие 100-балльной и 5-балльной шкал

Сумма баллов по 100-балльной шкале	Оценка по 5-балльной шкале
100-85	отлично

84-70	хорошо
69-50	удовлетворительно
49 и менее	неудовлетворительно

Критерии оценивания решения компетентностно-ориентированной задачи (нижеследующие критерии оценки являются примерными и могут корректироваться):

6-5 баллов выставляется обучающемуся, если решение задачи демонстрирует глубокое понимание обучающимся предложенной проблемы и разностороннее ее рассмотрение; свободно конструируемая работа представляет собой логичное, ясное и при этом краткое, точное описание хода решения задачи (последовательности (или выполнения) необходимых трудовых действий) и формулировку доказанного, правильного вывода (ответа); при этом обучающимся предложено несколько вариантов решения или оригинальное, нестандартное решение (или наиболее эффективное, или наиболее рациональное, или оптимальное, или единственно правильное решение); задача решена в установленное преподавателем время или с опережением времени.

4-3 балла выставляется обучающемуся, если решение задачи демонстрирует понимание обучающимся предложенной проблемы; задача решена типовым способом в установленное преподавателем время; имеют место общие фразы и (или) несущественные недочеты в описании хода решения и (или) вывода (ответа).

2-1 балла выставляется обучающемуся, если решение задачи демонстрирует поверхностное понимание обучающимся предложенной проблемы; осуществлена попытка шаблонного решения задачи, но при ее решении допущены ошибки и (или) превышено установленное преподавателем время.

0 баллов выставляется обучающемуся, если решение задачи демонстрирует непонимание обучающимся предложенной проблемы, и (или) значительное место занимают общие фразы и голословные рассуждения, и (или) задача не решена.