Государственное образовательное учреждение
«Приднестровский государственный университет им. Т.Г.Шевченко»
Инженерно-технический институт

Кафедра «Информационные технологии и автоматизированное управление производственными процессами»

УТВЕРЖДАЮ
Заведующий кафедрой ИТиАУПП

_____ Ю.А.Столяренко

«28» августа 2020 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
по дисциплине

Б1.О.04 «АРХИТЕКТУРА ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»

Направление подготовки
2.09.04.01 «Информатика и вычислительная техника»

Профиль (специализация) подготовки
Информационное и программное обеспечение вычислительных систем

Квалификация (степень) выпускника: **магистр**

Форма обучения: **очная, заочная**

Год набора: **2020 г.**

Разработал: доцент
А.Ю. Долгов

«28» августа 2020 г.

Тирасполь, 2020

Паспорт фонда оценочных средств по учебной дисциплине.

1. В результате изучения «Архитектура параллельных вычислительных систем» у обучающихся должны быть сформированы следующие компетенции:

Категория (группа) компетенций	Код и наименование	Код и наименование индикатора достижения универсальной компетенции
Общепрофессиональные компетенции и индикаторы их достижения		
	ОПК-5. Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем	ИД-1$_{ОПК-5}$ Знать современное программное и аппаратное обеспечение информационных и автоматизированных систем
		ИД-2$_{ОПК-5}$ Уметь модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем для решения профессиональных задач;
		ИД-3$_{ОПК-5}$ Иметь навыки разработки программного и аппаратного обеспечения информационных и автоматизированных систем для решения профессиональных задач
	ОПК-6. Способен разрабатывать компоненты программно-аппаратных комплексов обработки информации и автоматизированного проектирования	ИД-1$_{ОПК-6}$ знать: аппаратные средства и платформы инфраструктуры информационных технологий, виды, назначение, архитектуру, методы разработки и администрирования программно-аппаратных комплексов объекта профессиональной деятельности
		ИД-2$_{ОПК-6}$ Уметь: анализировать техническое задание, разрабатывать и оптимизировать программный код для решения задач обработки информации и автоматизированного проектирования
		ИД-3$_{ОПК-6}$ Владеть: навыками составления технической документации по использованию и настройке компонентов программно-аппаратного комплекса

2. Программа оценивания контролируемой компетенции:

Текущая аттестация	Контролируемые модули, разделы (темы) дисциплины и их наименования	Код контролируемой компетенции или его части	Наименование оценочного средства
РУБЕЖНЫЙ КОНТРОЛЬ	Архитектуры параллель-ных вычислительных систем	ОПК-5,ОПК-6	Т1, ЛР1
	Вычислительные системы на основе графических процессоров	ОПК-5,ОПК-6	Т1, Т2, ЛР2
РУБЕЖНАЯ АТТЕСТАЦИЯ	Распределенные вычислительные системы	ОПК-5,ОПК-6	Т2, ЛР3, ЛР4
Промежуточная аттестация		Код контролируемой компетенции или его части	Наименование оценочного средства
		ОПК-5, ОПК-6	экзамен

3. Показатели и критерии оценивания компетенции по этапам формирования, описание шкал оценивания

Этапы оценивания компетенции	Показатели достижения заданного уровня освоения компетенции	Критерии оценивания результатов обучения			
		2	**3**	**4**	**5**
Первый этап	ИД-1$_{ОПК-5}$ **Знать** современное программное и аппаратное обеспечение информационных и автоматизированных систем	Не знает	Знает основные понятия теории, но не знает способы применения современного программного и аппаратного обеспечения информационных и автоматизированных систем в профессиональной деятельности	Знает основные понятия и основы теории, но не может применять один из методов современного программного или аппаратного обеспечения информационных и автоматизированных систем в профессиональной деятельности	Знает основные понятия и основы теории. Умеет применять современное программное и аппаратное обеспечение информационных и автоматизированных систем
Второй этап	ИД-2$_{ОПК-5}$ **Уметь** модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем для решения профессиональных задач	Не умеет	Правильно определяет профессиональные задачи, но не умеет модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем для решения профессиональных задач	Умеет модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем для решения профессиональных задач, но не умеет обрабатывать результаты	Умеет модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем для решения профессиональных задач
Третий этап	ИД-3$_{ОПК-5}$ **Владеть** навыками разработки программного и аппаратного обеспечения информационных и автоматизированных систем для решения профессиональных задач	Не владеет	Владеет навыками разработки либо программного, либо аппаратного обеспечения информационных и автоматизированных систем для решения профессиональных задач, но не владеет порядком оформления технической документации	Владеет навыками разработки либо программного, либо аппаратного обеспечения информационных и автоматизированных систем для решения профессиональных задач	Владеет навыками разработки программного и аппаратного обеспечения информационных и автоматизированных систем для решения профессиональных задач

Первый этап	ИД-1$_{ОПК-6}$ **Знать:** аппаратные средства и платформы инфраструктуры информационных технологий, виды, назначение, архитектуру, методы разработки и администрирования программно-аппаратных комплексов объекта профессиональной деятельности	Не знает	Знает основные понятия теории, но не знает способы применения аппаратных средств и платформ инфраструктуры информационных технологий, виды, назначение, архитектуру, методы разработки и администрирования программно-аппаратных комплексов объекта профессиональной деятельности	Знает основные понятия и основы теории аппаратных средств и платформ инфраструктуры информационных технологий, виды, назначение, архитектуру, методы разработки и администрирования программно-аппаратных комплексов объекта профессиональной деятельности	Знает аппаратные средства и платформы инфраструктуры информационных технологий, виды, назначение, архитектуру, методы разработки и администрирования программно-аппаратных комплексов объекта профессиональной деятельности
Второй этап	ИД-2$_{ОПК-6}$ **Уметь:** анализировать техническое задание, разрабатывать и оптимизировать программный код для решения задач обработки информации и автоматизированнн ого проектирования	Не умеет	Правильно анализирует техническое задание, но не умеет разрабатывать и оптимизировать программный код для решения задач обработки информации и автоматизированнн ого проектирования	Умеет анализировать техническое задание и разрабатывать программный код для решения задач обработки информации и автоматизированног о проектирования, но не умеет их оптимизировать.	Умеет анализировать техническое задание, разрабатывать и оптимизировать программный код для решения задач обработки информации и автоматизированного проектирования
Третий этап	ИД-3$_{ОПК-6}$ **Владеть:** навыками составления технической документации по использованию и настройке компонентов программно-аппаратного комплекса	Не владеет	Владеет навыками составления технической документации по использованию компонентов программно-аппаратного комплекса, но не владеет навыками их настройки	Владеет навыками составления технической документации по использованию и настройке компонентов либо программной, либо аппаратной составляющей комплекса	Владеет навыками составления технической документации по использованию и настройке компонентов программно-аппаратного комплекса

4. Шкала оценивания

Согласно Положению «О порядке организации аттестации в ИТИ ПГУ им. Т.Г. Шевченко, итоговая оценка представляет собой сумму баллов, полученных студентом по итогу освоения дисциплины (модуля):

Оценка в традиционной шкале	Оценка в 100-балльной шкале	Буквенные эквиваленты оценок в шкале ЗЕ (% успешно аттестованных)
5 (отлично)	88–100	A (отлично) – 88-100 баллов
4 (хорошо)	70–87	B (очень хорошо) – 80-87баллов
		C (хорошо) – 70-79 баллов

3 (удовлетворительно)	50–69	D(удовлетворительно) – 60-69 баллов
		E(посредственно) – 50-59 баллов
2 (неудовлетворительно)	0–49	Fx– неудовлетворительно, с возможной пересдачей – 21-49 баллов
		F– неудовлетворительно, с повторным изучением дисциплины – 0-20 баллов

Расшифровка уровня знаний, соответствующего полученным баллам, дается в таблице, указанной ниже

A	“Отлично” - теоретическое содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному.
B	“Очень хорошо” - теоретическое содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом в основном сформированы, все предусмотренные программой обучения учебные задания выполнены, качество выполнения большинства из них оценено числом баллов, близким к максимальному.
C	“Хорошо” - теоретическое содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные программой обучения учебные задания выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые виды заданий выполнены с ошибками.
D	“Удовлетворительно” - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий, возможно, содержат ошибки.
E	“Посредственно” - теоретическое содержание курса освоено частично, некоторые практические навыки работы не сформированы, многие предусмотренные программой обучения учебные задания не выполнены, либо качество выполнения некоторых из них оценено числом баллов, близким к минимальному.
FX	“Условно неудовлетворительно” - теоретическое содержание курса освоено частично, необходимые практические навыки работы не сформированы, большинство предусмотренных программой обучения учебных заданий не выполнено, либо качество их выполнения оценено числом баллов, близким к минимальному; при дополнительной самостоятельной работе над материалом курса возможно повышение качества выполнения учебных заданий.
F	“Безусловно неудовлетворительно” - теоретическое содержание курса не освоено, необходимые практические навыки работы не сформированы, все выполненные учебные задания содержат грубые ошибки, дополнительная самостоятельная работа над материалом курса не приведет к какому-либо значимому повышению качества выполнения учебных заданий.

5. **Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций при изучении учебной дисциплины в процессе освоения образовательной программы**

5.1 Типовой вариант задания на контрольную работу

5.2. Типовой вариант задания на практическую работу

5.3 Типовой тест промежуточной аттестации

1. Какого рода ускорение происходит при конвейерной обработке?

1) Ускорение при конвейерной обработке происходит за счет снижения использования аппаратных сегментов для данного набора процессов, каждый из которых применяет эти ресурсы случайно выбранным способом. Каждый процесс использует разные ресурсы, и как только один ресурс освобождается данным процессором, он может быть использован следующим процессором, ожидая, пока предыдущий процесс достигнет конца конвейера.

+2) Ускорение при конвейерной обработке происходит за счет улучшения использования аппаратных ресурсов для заданного набора процессов, каждый из которых применяет эти ресурсы заранее предусмотренным способом. Каждый процесс использует одни и те же ресурсы, и как только некоторый ресурс освобождается данным процессом, он сразу же может быть использован следующим процессом, не ожидая, пока предыдущий процесс достигнет конца конвейера.

3) Ускорение при конвейерной обработке происходит за счет использования программных модулей для заданного набора процессоров, каждый из которых изменяет ресурсы заранее не предусмотренным образом. Один процесс использует одни ресурсы, и другой процесс использует другие ресурсы, поэтому они не связаны и не мешают друг другу выполнять наборы команд ожидая, пока все процессы достигнут конца конвейера.

2. Поясните понятие суперкомпьютера.

1) Суперкомпьютерами называют большие наборы отдельных устройств, реализованных на высокопроизводительных процессорах. В качестве основной характеристики таких компьютеров используется показатель производительности – величина, показывающая, какое количество арифметических операций он может выполнить за единицу времени.

2) Суперкомпьютерами называют устройства, реализующие высокоскоростные последовательные вычисления. Основной характеристикой таких компьютеров является эффективность – средняя доля времени выполнения алгоритма, в течение которой процессоры реально используются для решения задачи.

+3) Суперкомпьютерами называют устройства, реализующие высокопроизводительные параллельные вычисления. В качестве основной характеристики таких компьютеров используется показатель производительности – величина, показывающая, какое количество арифметических операций он может выполнить за единицу времени. К суперкомпьютерам относят лишь те компьютерные системы, которые имеют максимальную производительность в настоящее время.

3. Приведите пример задачи, обязательно требующей применения высокопроизводительных вычислений.

+1) Автомобилестроение, нефте- и газодобыча, фармакология, прогноз погоды и моделирование изменения климата, сейсморазведка, проектирование электронных устройств, синтез новых материалов.

2) Ракетостроение, добыча редких металлов, физиотерапия, астрономический прогноз и моделирование звездных величин, геологоразведка, проектирование электромеханических устройств, оргсинтез.

3) Кораблестроение, добыча полезных ископаемых, наркология, прогноз итогов выборов и моделирование исторических процессов, разведка недр, программирование новых систем, решение нелинейных уравнений.

4. Возможно ли увеличение производительности суперкомпьютера прямо пропорционально увеличению количества процессорных элементов?

1) Да, возможно, т.к. в реальной жизни накладные расходы по времени не влияют на коммуникации, поэтому увеличение производительности с ростом числа процессоров будет пропорционально их числу.

+2) Нет, невозможно, т.к. в реальной жизни очень много накладных расходов по времени приходится на коммуникации, поэтому увеличение производительности с ростом числа процессоров будет непропорционально меньше их числа.

3) Да, возможно, т.к. в реальной жизни очень мало накладных расходов по времени приходится на коммуникации, поэтому увеличение производительности с ростом числа процессоров будет непропорционально больше их числа.

5. В чем заключаются основные способы достижения параллелизма?

1) Основной способ достижения параллелизма заключаются в использовании программных модулей для заданного набора процессоров, каждый из которых изменяет ресурсы заранее не предусмотренным образом.

Один процесс использует одни ресурсы, и другой процесс использует другие ресурсы, поэтому они не связаны и не мешают друг другу выполнять наборы команд ожидая, пока все процессы достигнут конца конвейера.

2) Основной способ достижения параллелизма заключаются во внесении изменений в структуру компьютеров на основе принципа конвейерной обработки данных, переходящей в раздельное (последовательное) выполнение нескольких действий. Также, необходимо наблюдать за независимостью функциональных узлов устройств, а также разнообразием элементов вычислительной системы.

+3) Основной способ достижения параллелизма заключаются во внедрении новых решений в архитектуру компьютеров на основе принципа параллельной обработки данных, воплощающий идею одновременного (параллельного) выполнения нескольких действий. Кроме того, необходимо соблюдать независимость функционирования отдельных устройств, а также избыточность элементов вычислительной системы.

6. Что положено в основу классификации Флинна?

1) В основу классификации Флинна положена параллельность высокопроизводительных вычислительных систем, которые различают по количеству потоков команд и количеству данных для каждого потока, являющееся классификационным признаком.

+2) В основу классификации Флинна положена систематика, в рамках которой основное внимание при анализе архитектуры вычислительных систем уделяется способам взаимодействия последовательностей (потоков) выполняемых команд и обрабатываемых данных. При этом количество потоков команд и количество данных для каждого потока является классификационным признаком.

3) В основу классификации Флинна положен способ соединения суперкомпьютеров, которые используются как единый вычислительный ресурс (коммуникации скрыты) и пользователь получает практически неограниченные вычислительные мощности, т.е. виртуальный распределенный вычислительный ресурс без привязки к конкретному географическому месту.

7. В чем состоит принцип разделения многопроцессорных систем на мультипроцессоры и мультикомпьютеры?

1) Многопроцессорные системы, в которых обеспечивается когерентность локальной кэш-памяти разных процессоров (CC-NUMA), системы, в которых обеспечивается общий доступ к локальной памяти разных процессоров без поддержки на аппаратном уровне когерентности кэш-памяти (NCC-NUMA), микропроцессоры, использующие только локальную кэш-память (COMA), векторные параллельные процессоры (PVP), симметричные мультипроцессоры (SMP).

2) Многопроцессорные системы, в которых обеспечивается латентность локальной кэш-памяти разных процессоров (CC), системы, в которых обеспечивается параллельный доступ к локальной памяти разных процессоров на аппаратном уровне кэш-памяти (NUMA), векторные процессоры (VP), симметричные мультипроцессоры (SMP).

+3) Многопроцессорные системы делятся на мультипроцессоры и мультикомпьютеры по принципу использования ресурсов памяти. Так мультипроцессоры – это один компьютер со многими процессорами, который использует единую (централизованную) общую память (UMA), а мультикомпьютеры – это относительно независимые компьютеры с распределенной памятью (NUMA).

8. Какие классы систем известны для мультипроцессоров?

+1) Системы, в которых обеспечивается когерентность локальной кэш-памяти разных процессоров (CC-NUMA), системы, в которых обеспечивается общий доступ к локальной памяти разных процессоров без поддержки на аппаратном уровне когерентности кэш-памяти (NCC-NUMA), микропроцессоры, использующие только локальную кэш-память (COMA), векторные параллельные процессоры (PVP), симметричные мультипроцессоры (SMP).

2) Мультикомпьютеры делятся на два класса: массивно-параллельных систем (MPP) и кластеров (clusters).

3) Системы, в которых обеспечивается латентность локальной кэш-памяти разных процессоров (CC), системы, в которых обеспечивается параллельный доступ к локальной памяти разных процессоров на аппаратном уровне кэш-памяти (NUMA), векторные процессоры (VP), симметричные мультипроцессоры (SMP).

9. Что представляет собой массивно-параллельный компьютер?

+1) Массивно-параллельный компьютер – это компьютерная платформа, в которой число процессоров может произвольно меняться, с учетом заранее заданной производительности и/или стоимости. В дальнейшем она позволяет наращивать вычислительные мощности в зависимости от решаемых задач.

2) Массивно-параллельный компьютер – это способ соединения суперкомпьютеров, которые используются как единый вычислительный ресурс (коммуникации скрыты) и пользователь получает практически неограниченные вычислительные мощности, т.е. виртуальный распределенный вычислительный ресурс без привязки к конкретному географическому месту.

3) Массивно-параллельный компьютер – это компьютер, который получил название по двум принципам, заложенным в архитектуре процессоров:

1. конвейерная организация обработки потока команд;

2. введение в систему команд набора векторных операций, которые позволяют оперировать с целыми массивами данных.

10. Что представляет собой векторно-конвейерный компьютер?

1) Векторно-конвейерный компьютер – это компьютерная платформа, в которой число процессоров может произвольно меняться, с учетом заранее заданной производительности и/или стоимости. В дальнейшем она позволяет наращивать вычислительные мощности в зависимости от решаемых задач.

+2) Векторно-конвейерный компьютер – это компьютер, который получил название по двум принципам, заложенным в архитектуре процессоров:

1. конвейерная организация обработки потока команд;

2. введение в систему команд набора векторных операций, которые позволяют оперировать с целыми массивами данных.

3) Векторно-конвейерный компьютер – это способ соединения суперкомпьютеров, которые используются как единый вычислительный ресурс (коммуникации скрыты) и пользователь получает практически неограниченные вычислительные мощности, т.е. виртуальный распределенный вычислительный ресурс без привязки к конкретному географическому месту.

11. В чем состоят особенности сетей передачи данных для кластеров?

1) Сети передачи данных для кластеров – это системы, в которых обеспечивается латентность локальной кэш-памяти разных процессоров (CC), системы, в которых обеспечивается параллельный доступ к локальной памяти разных процессоров на аппаратном уровне кэш-памяти (NUMA), векторные процессоры (VP), симметричные мультипроцессоры (SMP).

2) Сети передачи данных для кластеров выполняются на основе коммутационных технологий, высокоскоростного сетевого оборудования и специального программного обеспечения, реализующего механизм передачи сообщений по стандартным сетевым протоколам, и позволяющего объединить в вычислительные системы компьютеры только одного типа, либо персональные компьютеры, либо мощные суперкомпьютеры.

+3) Сети передачи данных для кластеров строятся на основе коммуникационных технологий, а именно, высокоскоростного сетевого оборудования и специального программного обеспечения, реализующего механизм передачи сообщений над стандартными сетевыми протоколами, и позволяющего объединить в единые вычислительные системы компьютеры самого разного типа, начиная от персональных компьютеров и заканчивая мощными суперкомпьютерами.

12. Что представляет собой концепция метакомпютинга?

+1) Метакомпьютинг – способ соединения суперкомпьютеров, которые используются как единый вычислительный ресурс (коммуникации скрыты) и пользователь получает практически неограниченные вычислительные мощности, т.е. виртуальный распределенный вычислительный ресурс без привязки к конкретному географическому месту.

2) Метакомпьютинг – системы, в которых обеспечивается латентность локальной кэш-памяти разных процессоров (CC), системы, в которых обеспечивается параллельный доступ к локальной памяти разных процессоров на аппаратном уровне кэш-памяти (NUMA), векторные процессоры (VP), симметричные мультипроцессоры (SMP).

3) Метакомпьютинг – это компьютерная платформа, в которой число процессоров может произвольно меняться, с учетом заранее заданной производительности и/или стоимости. В дальнейшем она позволяет наращивать вычислительные мощности в зависимости от решаемых задач.

13. Каковы причины появления Grid проектов?

1) Структура Grid – это постоянное развитие вычислительных «облачных» технологий, стандартизация вычислительных процессов, превращение ресурса в услугу аутсорсинга, развитие сервисных отношений между заказчиком и поставщиком ИТ-услуг.

+2) В связи с необходимостью подключения к вычислительным системам компьютеров, расположенных далеко друг от друга, и настолько слабыми интернет-каналами, что доступность того или иного из них в произвольный момент времени не гарантирована, накладывает дополнительные требования на управление ресурсами. Структура Grid – это виртуальная организация, образованная над пространством реальных компьютеров, сетей, административных зон.

3) Структура Grid появилась в конце 80-х годов XX века как естественное продолжение платформы параллельного метекомпьютинга, дающего возможность достаточно легко соединять различные компьютерные и коммуникационные сегменты в единую сеть. Доступность того или иного из сегментов в произвольный момент времени не гарантирована, это накладывает дополнительные требования на управление ресурсами.

14. Каковы этапы численного эксперимента?

+1) При проведении численного эксперимента выделяют следующие этапы: построение математической модели исследуемого объекта, по которой выбирается численный метод, создание откомпилированного программного продукта, который в свою очередь после запуска и отладки выдаст искомый результат вычислений.

2) При проведении численного эксперимента выделяют следующие этапы: проведение регрессионного анализа исследуемого объекта, после которого выбирается численный метод, создание ассемблированного программной системы, которая после отладки и запуска выдаст предварительный результат вычислений.

3) При проведении численного эксперимента выделяют следующие этапы: построение стохастической модели исследуемого объекта, по которой предлагается числовой метод, создание компилированного программного средства, которое в свою очередь после запуска и отладки выдаст промежуточный результат вычислений.

15. Как можно определить требуемую производительность для решения конкретной задачи?

1) Требуемую производительность для решения конкретной задачи можно определить с помощью теста EPM, который решает системы дифференциальных алгебраических уравнений, количество которых и метод решения не меняются в зависимости от конкретной задачи, а при этом еще ощущаются недостатки реализации недостающих интерфейсов.

+2) Требуемую производительность для решения конкретной задачи можно определить с помощью теста Linpack, который решает системы линейных алгебраических уравнений, количество которых и метод решения варьируются в зависимости от конкретной задачи, при этом пользователем предоставляется реализация недостающих интерфейсов.

3) Требуемую производительность для решения конкретной задачи можно определить с помощью теста SVP, который решает системы нелинейных уравнений, качество которых и методика написания меняются в зависимости от условий постановки задачи, при этом пользователем предоставляется возможность реализовать свои собственные интерфейсы.

16. Как определяется модель «операция – операнды»?

1) Модель «операция – операнды» определяется следующим образом. Расписание для распределения вычислений между процессорами состоит в следующем: p – количество процессоров, используемых для выполнения алгоритма, тогда для параллельного выполнения вычислений необходимо задать множество – расписание:

$$Hp = \{(i, P_i, t_i) : i \in V\},$$

в котором для каждой операции $i \in V$ указывается номер используемого для выполнения операции процессора P_i и время начала выполнения операции t_i.

2) Модель «операции-операнды» определяется следующим образом. Оценку T_∞ рассматривают как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров

$$T_\infty = \min_{p \geq 1} T_p$$

+3) Модель «операция – операнды» определяется следующим образом. Представим множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного графа

$G=(V, R)$

где $V=\{1, \ldots, |V|\}$ есть множество вершин графа, представляющих выполняемые операции алгоритма, а R есть множество дуг графа (при этом дуга $r=(i, j)$ принадлежит графу только в том случае, если операция j использует результат выполнения операции i).

17. Как определяется расписание для распределения вычислений между процессорами?

+1) Расписание для распределения вычислений между процессорами состоит в следующем: p – количество процессоров, используемых для выполнения алгоритма, тогда для параллельного выполнения вычислений необходимо задать множество – расписание:

$$Hp = \{(i, P_i, t_i) : i \in V\},$$

в котором для каждой операции $i \in V$ указывается номер используемого для выполнения операции процессора P_i и время начала выполнения операции t_i.

2) Расписание для распределения вычислений между процессорами определяется следующим образом. Представим множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного графа

$G=(V, R)$

где $V=\{1, \ldots, |V|\}$ есть множество вершин графа, представляющих выполняемые операции алгоритма, а R есть множество дуг графа (при этом дуга $r=(i, j)$ принадлежит графу только в том случае, если операция j использует результат выполнения операции i).

3) Расписание для распределения вычислений между процессорами рассматривают как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров

$$T_\infty = \min_{p \geq 1} T_p$$

18. Как определяется время выполнения параллельного алгоритма?

1) Оценку T_∞ рассматривают как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров

$$T_\infty = \min_{p \geq 1} T_p$$

+2) Время выполнения параллельного алгоритма определяется максимальным значением времени, используемым в расписании:

$$T_p(G, H_p) = \max_{i \in V}(t_i + 1).$$

3) Оценки $T_p(G, H_p)$, $T_p(G)$ и T_p могут быть использованы в качестве показателей времени выполнения параллельного алгоритма.

19. Как определить минимально возможное время решения задачи?

1) Время выполнения параллельного алгоритма определяется максимальным значением времени, используемым в расписании:

$$T_p(G, H_p) = \max_{i \in V}(t_i + 1).$$

2) Оценки $T_p(G, H_p)$, $T_p(G)$ и T_p могут быть использованы для определения минимально возможного времени решения задачи.

+3) Оценку T_∞ рассматривают как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров

$$T_\infty = \min_{p \geq 1} T_p$$

20. Что понимается под паракомпьютером и для чего может оказаться полезным данное понятие?

+1) Паракомпьютером называется концепция вычислительной системы с бесконечным количеством процессоров, которая широко используется при теоретическом анализе параллельных вычислений.

2) Паракомпьютером называется способ вычисления больших величин в рамках системы с конечным количеством процессоров, которая используется при творческом синтезе параллельных вычислений.

3) Паракомпьютером называется концентрация вычислительной мощности системы с разветвленной процессорной структурой, которая достаточно часто используется при практическом анализе последовательных вычислений.

21. Какие оценки следует использовать в качестве характеристики времени последовательного решения задачи?

1) Оценки $T_g(F, Q_g)$, $T_g(K)$ и T_g могут быть использованы в качестве показателей времени выполнения параллельного алгоритма.

+2) Оценки $T_p(G, H_p)$, $T_p(G)$ и T_p могут быть использованы в качестве показателей времени выполнения параллельного алгоритма.

3) Оценки T_∞ и T_1 рассматривают как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров.

22. Как определить минимально возможное время параллельного решения задачи?

1) Время выполнения параллельного алгоритма определяется минимально возможным значением времени, используемым в расписании:

$$T_p(G, H_p) = \max_{i \in V}(t_i + 1).$$

2) Оценки T_∞ и T_1 рассматривают как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров.

+3) Оценку $T\infty$ можно рассматривать как минимально возможное время выполнения параллельного алгоритма при использовании неограниченного количества процессоров

$$T_\infty = \min_{p \geq 1} T_p$$

23. Как формулируется закон Густавсона-Барсиса?

+1) Закон Густавсона-Барсиса – это оценка максимально достижимого ускорения выполнения параллельной программы, в зависимости от количества одновременно выполняемых потоков вычислений («процессоров») и доли последовательных расчётов. Выражается формулой:

$$S_p = g + (1-g)p = p + (1-h)g,$$

где g – доля последовательных расчётов в программе, p – количество процессоров. Данную оценку ускорения называют **ускорением масштабирования**, так как данная характеристика показывает, насколько эффективно могут быть организованы параллельные вычисления при увеличении сложности решаемых задач.

2) Закон Густавсона-Барсиса иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей. Согласно этому закону, ускорение выполнения программы за счёт распараллеливания её инструкций на множестве вычислителей ограничено временем, необходимым для выполнения её последовательных инструкций:

$$S_p \leq \frac{1}{f + (1-f)/p} \leq S^* = \frac{1}{f}.$$

3) Закон Густавсона-Барсиса определяется следующим образом. Представим множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного графа

$G=(V,R)$

где $V=\{1,\ldots,|V|\}$ есть множество вершин графа, представляющих выполняемые операции алгоритма, а R есть множество дуг графа (при этом дуга $r=(i,j)$ принадлежит графу только в том случае, если операция j использует результат выполнения операции i).

24. Как определяется понятие ускорения?

1) Ускорение (speedup) использования параллельным алгоритмом процессоров при решении задачи определяется соотношением

$$E_p(n) = T_1(n) / (pT_p(n)) = S_p(n) / p.$$

Величина ускорения определяет среднюю долю времени выполнения алгоритма, в течение которой процессоры реально используются для решения задачи.

+2) Ускорение (speedup), получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений определяется величиной:

$$S_p(n) = T_1(n) / T_p(n),$$

т.е. как отношение времени решения задач на скалярной ЭВМ к времени выполнения параллельного алгоритма (величина *n* используется для параметризации вычислительной сложности решаемой задачи и может пониматься, например, как количество входных данных задачи).

3) Ускорение (speedup) определяется использованием параллельного алгоритма процессоров при решении задачи определяется соотношением

$$S_p \le \frac{1}{f + (1-f)/p} \le S^* = \frac{1}{f}.$$

Величина ускорения определяет массовую долю времени выполнения процесса, в течение которой процесс реально используют для решения задачи.

25. Как определяется понятие эффективности?

1) Эффективность (efficiency), получаемая при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений определяется величиной:

$$S_p(n) = T_1(n) / T_p(n),$$

т.е. как отношение времени решения задач на скалярной ЭВМ к времени выполнения параллельного алгоритма (величина *n* используется для параметризации вычислительной сложности решаемой задачи и может пониматься, например, как количество входных данных задачи).

+2) Эффективность (efficiency) использования параллельным алгоритмом процессоров при решении задачи определяется соотношением

$$E_p(n) = T_1(n) / (pT_p(n)) = S_p(n) / p.$$

Величина эффективности определяет среднюю долю времени выполнения алгоритма, в течение которой процессоры реально используются для решения задачи.

3) Эффективность (efficiency) использования параллельным алгоритмом процессоров при решении задачи определяется соотношением

$$S_p \le \frac{1}{f + (1-f)/p} \le S^* = \frac{1}{f}.$$

Величина эффективности определяет массовую долю времени выполнения процесса, в течение которой процесс реально используют для решения задачи.

26. Возможно ли достижение сверхлинейного ускорения?

1) Нет, невозможно. Ускорение не может оказаться больше числа используемых процессоров $S_p(n) < p$ – в этом случае говорят о существовании нелинейного (nonlinear) ускорения. Это противоречит закону Амдала, что в свою очередь приводит к парадоксу Бейзеля, т.к. ускорение стремится превысить число процессоров, но на практике сверхлинейное ускорение не может иметь место.

2) Да, теоретически возможно при определённых обстоятельствах. Ускорение может оказаться больше числа используемых процессоров $S_p(n) > p$ – только в идеальном случае и тогда говорят о существовании супервихревого (supersquere) ускорения. Благодаря тому, что эффективность превышает число процессоров, в идеале сверхлинейное ускорение может иметь место.

+3) При определённых обстоятельствах ускорение может оказаться больше числа используемых процессоров $S_p(n) > p$ – в этом случае говорят о существовании сверхлинейного (superlinear) ускорения. Несмотря на то, что ускорение превышает число процессоров, на практике сверхлинейное ускорение может иметь место.

27. Как определяется понятие стоимости вычислений?

+1) Стоимость вычислений определяется как произведение времени параллельного решения задачи и числа используемых процессоров: $C_p=pT_p$.

2) Стоимость вычислений определяется как множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в единицу времени $G=(V,R)$.

3) Стоимость вычислений определяется как ограничение роста производительности вычислительной системы с увеличением количества вычислителей. Согласно этому закону, ускорение выполнения программы за счёт распараллеливания её инструкций на множестве вычислителей ограничено временем, необходимым для выполнения её последовательных инструкций:

$$S_p \le \frac{1}{f+(1-f)/p} \le S^* = \frac{1}{f}.$$

28. Как формулируется закон Амдала?

1) Закон Амдала определяется следующим образом. Представим множество операций, выполняемых в исследуемом алгоритме решения вычислительной задачи, и существующие между операциями информационные зависимости в виде ациклического ориентированного графа

$G=(V,R)$

где $V=\{1,\ldots,|V|\}$есть множество вершин графа, представляющих выполняемые операции алгоритма, а R есть множество дуг графа (при этом дуга $r=(i,j)$ принадлежит графу только в том случае, если операция j использует результат выполнения операции i).

+2) Закон Амдала иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей. Согласно этому закону, ускорение выполнения программы за счёт распараллеливания её инструкций на множестве вычислителей ограничено временем, необходимым для выполнения её последовательных инструкций:

$$S_p \le \frac{1}{f+(1-f)/p} \le S^* = \frac{1}{f}.$$

3) Закон Амдала – это оценка максимально достижимого ускорения выполнения параллельной программы, в зависимости от количества одновременно выполняемых потоков вычислений («процессоров») и доли последовательных расчётов. Выражается формулой:

$$S_p = g + (1-g)p = p + (1-h)g,$$

где g – доля последовательных расчётов в программе, p – количество процессоров.

29. Как оценивается производительность суперкомпьютеров на тесте Linpack?

1) Тест Linpack – это компьютерная программа, которая решает довольно большое семейство нелинейных алгебраических уравнений. Решение основано на корреляционной зависимости, граф генерируется с помощью случайного генератора. Программа состоит из двух тестов. Первый тест в Linpack измеряет производительность двух процедур: DGEFA(SGEFA) и DGESL(SGESL) для 64- и 32-битных версий соответственно. Второй тест работает с матрицей размерности 1000.

2) Тест состоит из компьютерных программ, каждая из которых решает множество линейных дифференциальных уравнений. Решение основано на Q-разложении, строка генерируется с помощью определенного генератора. Программа состоит из двух тестов. Первый тест работает с матрицей размерности 1000. Второй тест, создан для проверки производительности «хорошо» распараллеленных вычислений.

+3) Тест представляет из себя компьютерную программу, которая решает плотную систему линейных алгебраических уравнений. Решение основано на LU-разложении, матрица генерируется с помощью псевдослучайного генератора. Программа состоит из трёх тестов. Первый тест в Linpack измеряет производительность двух процедур: DGEFA(SGEFA) и DGESL(SGESL) для 64- и 32-битных версий соответственно. Второй тест работает с матрицей размерности 1000. Третий тест, создан для проверки производительности «хорошо» распараллеленных вычислений.

30. Какова архитектура суперкомпьютеров BlueGene фирмы IBM?

1) Суперкомпьютер BlueGene можно называть кластером кластеров так как он представляет собой три объединенных модуля по 60 шасси BladeCenter H с установленными блэйд-модулями TriBlades, всего 180 TriBlades. Все TriBlades подсоединены к 288-портовому маршрутизатору Voltaire ISR2012 Infiniband. Каждый объединённый модуль также подсоединён к файловой системе Panasas через 12 серверов System x3755.

2) **BlueGene** является массово-параллельным суперкомпьютером. В этой модели используется сетевой коммутатор SeaStar2 с топологией трехмерного тора, основанный на процессоре PowerPC 400, версия сокета под двух-ядерные процессоры AMD Opteron – Socket AM2, а также микросхемы памяти DDR2 с 240 пинами на разъеме. В XT4 также имеется поддержка FPGA-сопроцессоров, которые можно добавлять как в сервисные блейды, так и в блейды, ответственные за ввод-вывод.

+3) Каждый чип Blue Gene состоит из четырёх процессорных ядер PowerPC 450 с тактовой частотой 850 МГц. Чип, 2 или 4 ГБ оперативной памяти и сетевые интерфейсы образуют вычислительный узел суперкомпьютера. 32 вычислительных узла объединяются в карту (Compute Node card), к которой можно подсоединить до 2 узлов ввода-вывода. Системная стойка вмещает в себя 32 таких карты.

Конфигурация Blue Gene с пиковой производительностью 1 петафлопс представляет собой 72 системные стойки, содержащие 294,912 процессорных ядер, объединённых в высокоскоростную оптическую сеть. Конфигурация Blue Gene может быть расширена до 216 стоек с общим числом процессорных ядер 884,736, чтобы достигнуть пиковую производительность в 3 петафлопса. В стандартной конфигурации системная стойка Blue Gene содержит 4,096 процессорных ядер.

31. Что такое параллелизм данных?

+1) Параллелизм данных (data parallel) обеспечивает распределение данных между всеми процессорами компьютера и операции над ними. Распределяемыми данными обычно являются массивы. Языки программирования, поддерживающие данную модель, допускают операции над массивами, позволяют использовать в выражениях целые массивы, вырезки из массивов. Распараллеливание операций над массивами, циклами обработки массивов позволяет увеличить производительность программы. Компилятор отвечает за генерацию кода, осуществляющего распределение элементов массивов и вычислений между процессорами. Каждый процессор отвечает за то подмножество элементов массива, которое расположено в его локальной памяти.

2) Параллелизм данных (data parallel) предполагает, что имеется последовательная программа, которая может исполняться независимо от большого числа различных исходных данных. Можно ввести параллелизм путём запуска некоторого числа этих программ параллельно. Ясно, что параллелизм данных может быть использован для почти линейного ускорения, если запуски требуют примерно одинакового времени. Т.к. не требуется никаких связей между процессами, то этот подход может быть эффективно использован на кластерах рабочих станций со слабой пропускной способностью сети.

3) В рамках подхода, основанного на параллелизме данных (data parallel), для каждой подзадачи пишется своя собственная программа на обычном языке программирования. Подзадачи должны обмениваться результатами своей работы, получать исходные данные. Практически такой обмен осуществляется вызовом процедур специализированной библиотеки. Контролируется распределение данных между различными процессорами и различными подзадачами, а также обмен данными.

32. Что такое параллелизм задач?

1) Параллелизм задач (message passing) обеспечивает распределение данных между всеми процессорами компьютера и операции над ними. Распределяемыми данными обычно являются массивы. Языки программирования, поддерживающие данную модель, допускают операции над массивами, позволяют использовать в выражениях целые массивы, вырезки из массивов. Распараллеливание операций над массивами, циклами обработки массивов позволяет увеличить производительность программы.

+2) В рамках подхода, основанного на параллелизме задач (message passing), для каждой подзадачи пишется своя собственная программа на обычном языке программирования. Подзадачи должны обмениваться результатами своей работы, получать исходные данные. Практически такой обмен осуществляется вызовом процедур специализированной библиотеки. Контролируется распределение данных между различными процессорами и различными подзадачами, а также обмен данными.

3) Параллелизм задач (message passing) предполагает, что имеется последовательная программа, которая может исполняться независимо от большого числа различных исходных данных. Можно ввести параллелизм путём запуска некоторого числа этих программ параллельно. Ясно, что параллелизм задач может быть использован для почти линейного ускорения, если запуски требуют примерно одинакового времени. Т.к. не

требуется никаких связей между процессами, то этот подход может быть эффективно использован на кластерах рабочих станций со слабой пропускной способностью сети.

33. На каком наборе операций базируется подход, основанный на параллелизме данных?

1) Подход, основанный на параллелизме данных, базируется на наборе парных и коллективных операций передачи данных, которые выполняются для процессов, принадлежащих одному и тому же коммуникатору. Коллективные операции применяются одновременно для всех процессов коммуникатора.

2) 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 операций, во многом совпадающих с типами данных в алгоритмических языках C и Fortran. Кроме того, в MPI имеются возможности для создания новых производных типов операций для более точного выполнения пересылаемых сообщений.

+3) На первом этапе используют встроенные в транслятор средства векторизации или распараллеливания. Второй этап работы с такой программой – анализ затрачиваемого времени разными частями программы и определение наиболее ресурсопотребляющих частей. Третий этап – замена алгоритма вычислений в наиболее критичных частях программы.

34. Когда применяется тривиальная декомпозиция?

1) Тривиальная декомпозиция применяется когда между процессорами распределяются различные задачи, она предполагает выполнение на каждом процессоре одной и той же задачи, но над разными наборами данных. Части данных первоначально распределены между процессорами, которые обрабатывают их, после чего результаты суммируются некоторым образом в одном месте (обычно на консоли кластера). Данные должны быть распределены так, чтобы объем работы для каждого процесора был примерно одинаков, то есть декомпозиция должна быть сбалансированной. В случае дисбаланса эффективность работы кластера может быть снижена.

2) При тривиальной декомпозиции исходная задача разбивается на ряд последовательных действий, которые могут быть выполнены на различных узлах кластера независимо от промежуточных результатов, но строго последовательно.

+3) Тривиальная декомпозиция предполагает, что имеется последовательная программа, которая может исполняться независимо от большого числа различных исходных данных. Можно ввести параллелизм путём запуска некоторого числа этих программ параллельно. Ясно, что тривиальный параллелизм может быть использован для почти линейного ускорения, если запуски требуют примерно одинакового времени. Т.к. не требуется никаких связей между процессами, то этот метод может быть эффективно использован на кластерах рабочих станций со слабой пропускной способностью сети.

35. Что такое фермы задач?

1) Фермы задач (*task farming)* обеспечивают распределение данных между всеми процессорами компьютера и операции над ними. Распределяемыми данными обычно являются массивы. Языки программирования, поддерживающие данную модель, допускают операции над массивами, позволяют использовать в выражениях целые массивы, вырезки из массивов. Фермы задач определяют операции над массивами, циклами обработки массивов позволяет увеличить производительность программы. Компилятор отвечает за генерацию кода, осуществляющего распределение элементов массивов и вычислений между процессорами. Каждый процессор отвечает за то подмножество элементов массива, которое расположено в его локальной памяти.

+2) Фермы задач (*task farming)* – один из видов функциональной декомпозиции. Этот термин обозначает способ организации многомашинной системы, при котором одна из машин действует как планировщик, а другие – как рабочие. Отдельная задача в функциональной декомпозиции может быть распределена между некоторым числом процессоров. Для обработки выбираются небольшие порции данных и один процессор, *источник работы*, или планировщик, управляет набором необработанных порций данных. Некоторое число *рабочих* процессоров периодически запрашивают порцию данных из источника, обрабатывают её и затем отправляют результаты к *сборщику* результатов, который может совпадать или нет с источником работы.

3) Фермы задач (*task farming)* предполагают, что имеется последовательная программа, которая может исполняться независимо от большого числа различных исходных данных. Можно ввести фермы задач путём запуска некоторого числа этих программ параллельно. Ясно, что фермы задач могут быть использованы для почти линейного ускорения, если запуски требуют примерно одинакового времени. Так как не требуется никаких

связей между процессами, то этот метод может быть эффективно использован на кластерах рабочих станций со слабой пропускной способностью сети.

36. Каковы преимущества программирования на MPI?

1) В MPI интерфейс программиста проще и примитивнее, а интерфейс пользователя сложнее и запутаннее, предусмотрено динамическое размножение ветвей, возможно взаимодействие приложений, запущенных одним и тем же пользователем, загрузчик приложений, создающий собственно Виртуальную Параллельную Машину, является постоянно запущенным (полезно для исследовательских целей); срок жизни загрузчика строго совпадает со сроком жизни одного конкретного приложения (более простое, надежное и компактное решение).

+2) MPI позволяет в значительной степени снизить остроту проблемы переносимости параллельных программ между разными компьютерными системами. Библиотеки MPI содействуют повышению эффективности параллельных вычислений, в максимальной степени учитывая возможности используемого компьютерного оборудования. MPI уменьшает сложность разработки параллельных программ, т.к., с одной стороны, большая часть основных операций передачи данных предусматривается стандартом MPI, а с другой стороны, уже имеется большое количество библиотек параллельных методов, созданных с использованием MPI.

3) Модель программирования в MPI через общую память с синхронизацией семафорами – это базовая функция для многопроцессорных компьютеров. Функции работы с Shared memory непосредственно входят в состав каждой многозадачной операционной системы. В пределах одного компьютера все остальные средства межпроцессовой коммуникации реализуются через SHM, и потому заведомо являются менее быстрыми.

37. Как описываются в MPI передаваемые сообщения?

1) Для передачи сообщения процесс-отправитель должен выполнить функцию:

MPI_Get_count (MPI_Status *status, MPI_Datatype type, int *count);

+2) Для передачи сообщения процесс-отправитель должен выполнить функцию:

int MPI_Send(void *buf, int count, MPI Datatype type, int dest, int tag, MPI Comm comm);

3) Для передачи сообщения процесс-отправитель должен выполнить функцию:

int MPI_Recv(void *buf, int count, MPI Datatype type, int source, int tag, MPI Comm comm, MPI Status *status);

38. В чем различие парных и коллективных операций передачи данных?

1) Под парной операцией в рамках MPI понимается множество одновременно выполняемых процессов, имеющих раздельные адресные пространства. Каждый процесс парной операции порождается на основе копии одного и того же программного кода (модель SPMP - одна программа множество процессов single program multiple processes). Коллективные операции передачи данных, представленные в виде исполняемой программы, должены быть доступены в момент запуска параллельной программы на всех используемых процессорах.

2) Количество процессов и число используемых процессоров определяется в момент запуска парных операций средствами среды исполнения MPI и в ходе вычислений меняться не может. В стандарте MPI предусматривается возможность применения коллективных операций, которые зависят от изменения количества процессов от 0 до p-1, где p есть общее количество процессов.

+3) Функции MPISend и MPIRecv обеспечивают возможность выполнения парных операций передачи данных между двумя процессами параллельной программы. Для выполнения коммуникационных коллективных операций, в которых принимают участие все процессы коммуникатора, в MPI предусмотрен специальный набор функций MPIBcast и MPIReduce.

39. Какая функция MPI обеспечивает передачу данных от одного процесса всем процессам?

+1) Функция MPI_Bcast осуществляет рассылку данных из буфера buf, содержащего count элементов типа type с процесса, имеющего номер root, всем процессам, входящим в коммуникатор comm. При этом следует отметить, что функция MPI_Bcast определяет коллективную операцию и, тем самым, при выполнении необходимых рассылок данных вызов функции MPI_Bcast должен быть осуществлен всеми процессами указываемого коммуникатора.

2) Функция MPI_Status выполняет чтение сообщения, некоторые параметры которого могут оказаться неизвестными, а именно: число считанных элементов, идентификатор сообщения и адрес отправителя. Переменные status должны быть явно объявлены в MPI-программе.В языке C status - это структура типа MPI_Status с тремя полями MPI_SOURCE, MPI_TAG, MPI_ERROR.

3) Функция MPI_REDUCE объединяет элементы входного буфера каждого процесса в группе, используя операцию op , и возвращает объединенное значение в выходной буфер процесса с номером root. Буфер ввода

определен аргументами sendbuf, count и datatype; буфер вывода определен параметрами recvbuf, count и datatype; оба буфера имеют одинаковое число элементов одинакового типа. Функция вызывается всеми членами группы с одинаковыми аргументами count, datatype, op, root и comm.

40. Что понимается под операцией редукции?

1) В общем случае операцией редукции называется операция, аргументом которой является вектор, а результатом – скалярная величина, полученная применением некоторой математической операции ко всем компонентам вектора. В частности, если компоненты вектора расположены в адресных пространствах процессов, выполняющихся на различных процессорах, то в этом случае говорят о глобальной (параллельной) редукции.

+2) Под операцией редукции данных понимаются операции, выполняемые над собираемыми значениями с одновременным осуществлением обработки данных. Операция редукции данных является примером часто выполняемой коллективной операции передачи данных от всех процессов одному процессу.

3) Операции редукции – это используемые средства для измерения времени работы программ, которое обычно зависит, от аппаратной платформы, операционной системы, алгоритмического языка и т.п. Стандарт MPI включает определение операций редукции для измерения времени, использование которых позволяет устранить зависимость от среды выполнения параллельных программ.

5.4 Вопросы к экзамену или зачету

1. Параллельные компьютеры и супер-ЭВМ.
2. Понятие высокопроизводительных вычислений.
3. Методы увеличения производительности.
4. Классификация компьютерных систем.
5. Детализация архитектур по достижимой степени параллелизма.
6. Векторно-конвейерные компьютеры.
7. Вычислительные системы с распределенной памятью (мультикомпьютеры).
8. Параллельные компьютеры с общей памятью (мультипроцессоры).
9. Концепция GRID и метакомпьютинг.
10. Численный эксперимент и параллельная форма алгоритма.
11. Схемы параллельного выполнения алгоритма.
12. Показатели эффективности параллельного алгоритма.
13. Оценка достижимого параллелизма. Законы Амдала.
14. Две парадигмы программирования.
15. Методология проектирования параллельных алгоритмов.
16. Декомпозиция для выделения параллелизма.
17. Концепция передачи сообщений.
18. Основные понятия и определения программирования на MPI.
19. Базовые функции MPI.
20. Коллективные операции передачи данных.
21. Двухточечный обмен сообщениями в MPI.
22. Стандартный обмен в MPI.
23. Синхронный блокирующий обмен в MPI.
24. Буферизованный обмен в MPI.
25. Обмен "по готовности" в MPI.
26. Подпрограммы-пробники в MPI.
27. Совместные прием и передача в MPI.
28. Проверка выполнения обмена в MPI.
29. Отложенные обмены в MPI.
30. Коллективный обмен данными в MPI.
31. Широковещательная рассылка в MPI.
32. Управление областью взаимодействия и группой процессов в MPI.
33. Группы процессов и создание групп процессов в MPI.
34. Управление коммуникаторами в MPI.
35. Операции обмена между группами процессов (интеробмен) в MPI.

36. Концепция параллельных данных.
37. Операции с параллельные данными.
38. Технология OpenMP.
39. Распределение вычислительной нагрузки между потоками в OpenMP.
40. Управление данными для параллельно-выполняемых потоков в OpenMP.
41. Определение общих и локальных переменных в OpenMP.
42. Распределение вычислительной нагрузки между потоками.
43. Расширенные возможности OpenMP.
44. Синхронизация состояния памяти.
45. Управление количеством потоков.
46. Управление вложенностью параллельных фрагментов.
47. Исчисление взаимодействующих систем и высокопроизводительные вычисления.
48. Математические конструкции CCS.
49. Поведение процессов CCS.
50. Формальное определение CCS.