ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ НАУКИ
ИНСТИТУТ ПРИКЛАДНЫХ МАТЕМАТИЧЕСКИХ ИССЛЕДОВАНИЙ
КАРЕЛЬСКОГО НАУЧНОГО ЦЕНТРА
РОССИЙСКОЙ АКАДЕМИИ НАУК
(ИПМИ КарНЦ РАН)

УТВЕРЖДАЮ
Директор ИПМИ КарНЦ РАН
д.ф.-м.н.
В.В. Мазалов
_____ 2014 г.

РАБОЧАЯ ПРОГРАММА
ДИСЦИПЛИНЫ

Методы и алгоритмы параллельных вычислений

Основной образовательной программы
профессионального образования (аспирантуры)
Направление подготовки:
09.06.01 Информатика и вычислительная техника
Профиль:
05.13.18 Математическое моделирование, численные методы
и комплексы программ
Квалификация: Исследователь. Преподаватель-исследователь.

Форма обучения
очная

Петрозаводск 2014

Составители рабочей программы
В.н.с., проф., д.ф.-м.н. _____ _____ Соколов А.В.
(должность, ученое звание, ученая степень) (подпись) (Ф.И.О.)

Рабочая программа утверждена на заседании Ученого совета ИПМИ КарНЦ РАН

«22» июня 2014 г., протокол № 6

Председатель Ученого совета
Д.ф.-м.н., проф. _____ В.В. Мазалов

1. Цели освоения учебной дисциплины «Методы и алгоритмы параллельных вычислений»:

- Сформировать базовое представление, знания, умения и навыки аспирантов по основам методов и алгоритмов параллельных вычислений для разработки программ на языках программирования С, С++ и др.
- Подготовить аспирантов к применению знаний параллельных методов и алгоритмов после окончания обучения в профессиональной деятельности.

2. Место дисциплины в структуре основной профессиональной образовательной программы послевузовского профессионального образования (аспирантура)

Дисциплина входит в раздел дисциплин по выбору образовательной компоненты ООП.

3. Перечень компетенций, которые должны быть сформированы у выпускника в результате освоения дисциплины "Методы и алгоритмы параллельных вычислений"

Выпускник, освоивший данную дисциплину, должен обладать следующими компетенциями: **универсальными компетенциями:**

готовностью участвовать в работе российских и международных исследовательских коллективов по решению научных и научно-образовательных задач (УК-3);

способностью планировать и решать задачи собственного профессионального и личностного развития (УК-6).

общепрофессиональными компетенциями:

владением методологией теоретических и экспериментальных исследований в области профессиональной деятельности (ОПК-1);

владением культурой научного исследования, в том числе с использованием современных информационно-коммуникационных технологий (ОПК-2);

профессиональными компетенциями:

способность к разработке эффективных параллельных вычислительных алгоритмов с применением современных компьютерных технологий (ПК-3);

готовность к реализации математического обеспечения в виде комплексов параллельных проблемно-ориентированных программ для проведения вычислительного эксперимента (ПК-4);

способность к разработке параллельного программного обеспечения и алгоритмов интерпретации эксперимента на основе его математической модели (ПК-7);

способность к разработке параллельных систем компьютерного и имитационного моделирования (ПК-8).

4. Требования к уровню подготовки аспиранта, завершившего изучение данной дисциплины

Аспиранты, завершившие изучение данной дисциплины, должны:

- знать: методы и алгоритмы параллельного программирования, тенденции и перспективы развития технологий параллельного программирования, современное состояние и принципиальные возможности параллельных архитектур и программных средств;

- уметь: использовать полученные знания для создания параллельных программ в различных предметных областях;

-владеть: приемами разработки параллельных программ на языках параллельного программирования.

5. Структура и содержание дисциплины "Методы и алгоритмы параллельных вычислений".

Общая трудоемкость дисциплины (модуля) составляет 2 зачетных единицы 72 часа аудиторных.

Вид учебной работы	Объем часов / зачетных единиц
Обязательная аудиторная учебная нагрузка (всего)	72 часа / 2 ЗЕ
В том числе:	
лекции	36 часов / 1 ЗЕ
семинары	36 часов / 1 ЗЕ
Практические занятия	
Самостоятельная работа	
Вид контроля по дисциплине	зачет

6. Темы и часы лекций и лабораторных занятий (семинаров)

№ темы	Название раздела/ темы	Лекций (час)	Семинаров (час)	Самост. работа (час)
1	Архитектуры современных процессоров. Многоядерность. Многопроцессорность. Использование графических процессоров в высокопроизводительных вычислениях. Распределенные системы	4	4	-
2	Моделирование и анализ параллельных вычислений. Модель вычислений в виде графа. Показатели эффективности. Масштабируемость. Закон Амдаля	4	4	-
3	Процессы и потоки. Семафоры. Мьютиксы. Мониторы. Классические задачи синхронизации. Производители-потребители. Читатели-писатели. Взаимоблокировка	4	4	-
4	Планирование параллельных заданий. Work-sharing и Work-stealing runtime scheduler. Структуры данных, используемые в Work-stealing. Ready-queues,Cactus-stacks	4	4	-
5	Языки и библиотеки параллельного программирования. Cilk, OpenCL, Go, T++, FPTL, Пифагор, CUDA, Parallel Java, C++ 11, Intel TBB,Microsoft TPL	4	4	-
6	Оптимизация параллелизма и локальности данных. Влияние кэш-памяти на работу параллельных программ. Cache-Oblivion структуры данных. Примеры: Использование Unrolled linked lists для реализации работы с несколькими очередями и стеками	4	4	-
7	Паралелльная реализация связных списков. Coarse-Grained Synchronization . Fine-Grained Synchronization, Optimistic Synchronization, Lazy Synchronization, Non-Blocking Synchronization	4	4	-

8	Паралелльная реализация FIFO-очередей. A Bounded Partial Queue, Unbounded Total Queue, Unbounded Lock-Free Queue, Memory Reclamation and the ABA Problem	4	4	-
9	Паралелльная реализация LIFO-стеков. Unbounded Lock-Free Stack, Elimination	4	4	-

Предполагается проведение научного семинара по книге [1]. Темы докладов- Lock-Base и Lock-Free реализация стеков, очередей, приоритетных очередей, хеш-таблиц, скип-списков, сортировки и т.п.

7. Содержание дисциплины:

Введение. Архитектуры современных процессоров. Многоядерность. Многопроцессорность. Использование графических процессоров в высокопроизводительных вычислениях. Распределенные системы.
Моделирование и анализ параллельных вычислений. Модель вычислений в виде графа. Показатели эффективности. Масштабируемость. Закон Амдаля.

Процессы и потоки. Семафоры. Мьютиксы. Мониторы. Классические задачи синхронизации. Производители-потребители. Читатели-писатели. Взаимоблокировка. Планирование параллельных заданий. Work-sharing и Work-stealing runtime scheduler. Структуры данных, используемые в Work-stealing. Ready-queues, Cactus-stacks.

Языки и библиотеки параллельного программирования. Cilk, OpenCL, Go, T++, FPTL, Пифагор, CUDA, Parallel Java, C++ 11, Intel TBB,Microsoft TPL. Оптимизация параллелизма и локальности данных. Влияние кэш-памяти на работу параллельных программ. Cache-Oblivion структуры данных. Примеры: Использование Unrolled linked lists для реализации работы с несколькими очередями и стеками.

Параллельные реализации структур данных. Паралелльная реализация связных списков. Coarse-Grained Synchronization. Fine-Grained Synchronization, Optimistic Synchronization, Lazy Synchronization, Non-Blocking Synchronization. Паралелльная реализация FIFO-очередей. A Bounded Partial Queue, Unbounded Total Queue, Unbounded Lock-Free Queue, Memory Reclamation and the ABA Problem. Паралелльная реализация LIFO-стеков. Unbounded Lock-Free Stack. Elimination.

8. Самостоятельная работа аспирантов:

Целями самостоятельной работы аспирантов являются полное и глубокое усвоение материала программы дисциплины, развитие навыков самообразования, навыков исследовательской работы и развитие познавательных способностей.

Для достижения этих целей используются в основном следующие виды работы: чтение и конспектирование рекомендуемых литературных источников, активный поиск электронных ресурсов в сети Интернет и отбор из них качественных документов, подготовка к участию в семинарах, включая подготовку докладов, получение консультаций у научного руководителя и других специалистов по вопросам изучаемой дисциплины.

9. Учебно-методическое и информационное обеспечение дисциплины «Методы и алгоритмы параллельных вычислений»

9.1. Основная литература

1. Соколов А.В., Барковский Е.А., Кучумов Р.И., Сазонов А. М. Методы и алгоритмы параллельных вычислений. ПетрГУ, 2015. 66 с.
2. Гергель, В. П. Теория и практика параллельных вычислений : учебное пособие / В. П. Гергель. - М. : Интернет-Университет информационных технологий, 2007. - 423 с. - (Основы информационных технологий). - Библиогр. : с. 418 - 423. - ISBN 978-5-9556-0096-3

3. http://www.cs.technion.ac.il/~erez/

9.2. Дополнительная литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002.
2. Некоторые алгоритмы планирования вычислений в многопроцессорных системах реального времени / Учреждение Рос. акад. наук Вычисл. центр им. А. А. Дородницына РАН ; [отв. ред. Л. Л. Вышинский]. - Москва : Вычислительный центр им. А. А. Дородницына РАН, 2010. - 73 с. : ил. ; 21 см. - Библиогр. в конце ст. - ISBN 978-5-91601-029-9

Вопросы к зачету

1. Архитектуры современных процессоров. Многоядерность. Многопроцессорность. Использование графических процессоров в высокопроизводительных вычислениях. Распределенные системы. Моделирование и анализ параллельных вычислений. Модель вычислений в виде графа.
2. Показатели эффективности. Масштабируемость. Закон Амдаля.
3. Процессы и потоки. Семафоры. Мьютиксы. Мониторы.
4. Классические задачи синхронизации. Производители-потребители. Читатели-писатели. Взаимоблокировка.
5. Планирование параллельных заданий. Work-sharing и Work-stealing runtime scheduler. Структуры данных, используемые в Work-stealing. Ready-queues,Cactus-stacks.
6. Языки и библиотеки параллельного программирования. Cilk, OpenCL, Go, T++, FPTL, Пифагор, CUDA, Parallel Java, C++ 11, Intel TBB,Microsoft TPL.
7. Оптимизация параллелизма и локальности данных. Влияние кэш-памяти на работу параллельных программ. Cache-Oblivion структуры данных. Примеры: Использование Unrolled linked lists для реализации работы с несколькими очередями и стеками.
8. Паралелльная реализация связных списков. Coarse-Grained Synchronization . Fine-Grained Synchronization, Optimistic Synchronization, Lazy Synchronization, Non-Blocking Synchronization.
9. Паралелльная реализация FIFO-очередей. A Bounded Partial Queue, Unbounded Total Queue, Unbounded Lock-Free Queue, Memory Reclamation and the ABA Problem.
10. Паралелльная реализация LIFO-стеков. Unbounded Lock-Free Stack, Elimination.

10. Перечень программного обеспечения

Транслятор с языков C и C++, поддерживающий технологии OpenMP и MPI.

11. Материально-техническое обеспечение дисциплины «Методы и алгоритмы параллельных вычислений»

При освоении дисциплины требуется:
Компьютерный класс для проведения лабораторных работ, имеющий необходимое количество рабочих мест, с набором базового программного обеспечения для разработки параллельных программ на языке программирования C++, с возможностью выхода на кластер КарНЦ РАН.

12. Особенности организации образовательного процесса для инвалидов и лиц с ограниченными возможностями здоровья

Обучение инвалидов и лиц с ограниченными возможностями здоровья осуществляется в соответствии с: ст.79, 273-ФЗ «Об образовании в Российской Федерации» 1. Раздел IV, п.п. 46-51 приказа Минобрнауки России от 19.11.2013 № 1259 «Об утверждении Порядка организации и осуществления образовательной деятельности по образовательным программам высшего образования – программам подготовки научно-педагогических кадров в аспирантуре (адъюнктуре)» 2. Методические рекомендации по организации образовательного

процесса для обучения инвалидов и лиц с ограниченными возможностями здоровья в образовательных организациях высшего образования, в том числе оснащенности образовательного процесса (утверждены заместителем Министра образования и науки РФ А.А.Климовым от 08.04.2014 г. № АК-44/05 вн)

11. Фонд оценочных материалов по дисциплине «Методы и алгоритмы параллельных вычислений»

1. В классификации ЭВМ Флинна системы MIMD – это системы с

1) Одиночным потоком команд и множественным потоком данных
2) Множественным потоком команд и множественным потоком данных
3) Одиночным потоком команд и одиночным потоком данных

2. Мультипроцессоры это –

1) Системы с распределенной памятью
2) Системы с общей разделяемой памятью
3) Многоядерные системы

3. Системы NUMA это –

1) Системы с однородным доступом к общей памяти
2) Системы с неоднородным доступом к общей памяти
3) Системы с распределенной памятью

4. Ускорение параллельного алгоритма это –

1) Время решения задачи на одном процессоре, умноженное на число процессоров
2) Отношение времени решения задачи на одном процессоре к времени решения задачи на n процессорах
3) Отношение времени решения задачи на n процессорах к времени решения задачи на одном процессоре

5. Ускорение параллельного алгоритма

1) Всегда равно числу процессоров
2) Всегда меньше числа процессоров
3) Может быть меньше, равно и больше числа процессоров

6. Как определяется модель параллельных вычислений "операция - операнды"?

1) В виде дерева вершины-операции, дуги - информационные зависимости
2) В виде графа вершины-операции, дуги - информационные зависимости
3) В виде графа вершины- операнды, дуги - операции

7. Как определяется расписание для распределения вычислений между процессорами?

1) В виде последовательности, в которой для каждой операции указывается номер используемого для выполнения операции процессора и время завершения выполнения операции

2) В виде последовательности, в которой для каждой операции указывается номер используемого для выполнения операции процессора и время начала выполнения операции
3) В виде последовательности, в которой для каждой операции указывается номер используемого для выполнения операции процессора и длительность выполнения операции

8. Как определяется время выполнения параллельного алгоритма?
1) Определяется минимальным значением времени, используемым в расписании
2) Определяется максимальным значением времени, используемым в расписании
3) Определяется последним значением времени, используемым в расписании

9. Как обеспечить минимальное время исполнения алгоритма
1) Выбором оптимального расписания
2) Выбором оптимальной вычислительной схемы
3) Выбором и того и другого

10. Что понимается под паракомпьютером и для чего может оказаться полезным данное понятие?
1) Концепция вычислительной системы с количеством процессоров равным двум, которая полезна для теоретической оценки возможного времени выполнения параллельного алгоритма на двухпроцессорной системе
2) Концепция вычислительной системы с бесконечным количеством процессоров, которая полезна для теоретической оценки минимально возможного времени выполнения параллельного алгоритма
3) Концепция вычислительной системы с одним процессором, которая полезна для теоретической оценки максимально возможного времени выполнения последовательного алгоритма

11. Какие оценки следует использовать в качестве характеристики времени последовательного решения задачи при оценке эффективности параллельного решения исследуемой задачи?
1) Оценку одного выбранного алгоритма решения задачи на одном процессоре
2) Оценку минимального возможного времени решения задачи на одном процессоре
3) Оценку максимального возможного времени решения задачи на одном процессоре

12. Как определить минимально возможное время параллельного решения задачи по графу "операнды - операции"?
1) Эта величина определяется длиной минимального пути вычислительной схемы алгоритма
2) Эта величина определяется длиной максимального пути вычислительной схемы алгоритма
3) Эта величина определяется суммой длин всех путей вычислительной схемы алгоритма

13. Какие зависимости могут быть получены для времени параллельного решения задачи при увеличении или уменьшения числа используемых процессоров?

1) При уменьшении числа используемых процессоров время выполнения алгоритма уменьшается пропорционально величине уменьшения количества процессоров

2) При уменьшении числа используемых процессоров время выполнения алгоритма увеличивается пропорционально величине уменьшения количества процессоров

3) При уменьшении числа используемых процессоров время выполнения алгоритма увеличивается пропорционально квадрату величины уменьшения количества процессоров

14. В каком из ответов описана каскадная схема суммирования?

1) На первой итерации каскадной схемы все исходные данные разбиваются на последовательности из n элементов, и для каждой последовательности вычисляется сумма их значений, далее все полученные суммы также разбиваются на последовательности из n элементов, и снова выполняется суммирование значений последовательностей т.д.
2) На первой итерации каскадной схемы все исходные данные разбиваются на две равных по числу элементов последовательности, и для каждой последовательности вычисляется сумма их значений, затем находится сумма этих двух чисел
3) На первой итерации каскадной схемы все исходные данные разбиваются на пары, и для каждой пары вычисляется сумма их значений, - далее все полученные суммы также разбиваются на пары, и снова выполняется суммирование значений пар и т.д.

15. В каком из ответов описан модифицированный вариант каскадной схемы суммирования?

1) На первом этапе вычислений все суммируемые значения подразделяются на (n / log 2 n) групп, в каждой из которых содержится log 2 n элементов; далее для каждой группы вычисляется сумма значений при помощи последовательного алгоритма суммирования; вычисления в каждой группе могут выполняться независимо друг от друга (т.е. параллельно – для этого необходимо наличие не менее (n / log 2 n) процессоров); − на втором этапе для полученных (n / log 2 n) сумм отдельных групп применяется обычная каскадная схема
2) На первой итерации каскадной схемы все исходные данные разбиваются на две равных по числу элементов последовательности, и для каждой последовательности вычисляется сумма их значений, затем находится сумма этих двух чисел
3) На первой итерации этой схемы все исходные данные разбиваются на пары, и для каждой пары вычисляется сумма их значений, - далее все полученные суммы также разбиваются на пары, и снова выполняется суммирование значений пар и т.д.

16. Как формулируется закон Амдаля?

1) Ускорение процесса вычислений при использовании p процессоров ограничивается величиной обратно-пропорциональной квадрату доли последовательных вычислений в применяемом алгоритме обработки данных
2) Ускорение процесса вычислений при использовании p процессоров ограничивается величиной обратно-пропорциональной доле последовательных вычислений в применяемом алгоритме обработки данных

17. Каким образом количество потоков влияет на эффективность выполнения параллельных программ?

1) При малом количестве потоков (меньшим, чем число ядер/процессоров) не будет задействован полностью потенциал компьютерного оборудования
2) При большом количестве потоков могут увеличиться затраты на их обслуживание – потоки надо создавать и завершать; при нехватке вычислительных устройств (ядер/процессоров) для потоков нужно прерывать выполнение активных потоков, сохранять их состояния и передавать на выполнение новые потоки; следует отметить,

что необходимое время на переключение потоков обычно является небольшим, а затраты на создание и завершение потоков можно снизить, если выполнять эти действия только при старте и завершении параллельной программ

3) Более серьезная причина потери эффективности при большом количестве потоков может состоять в ухудшении продуктивного использования кэш-памяти; как известно, кэш-память является во много (10-100) раз быстрее обычной оперативной памяти и быстродействие программы можно значительно повысить, если обеспечить присутствие обрабатываемых данных в кэш- памяти; в ситуациях, когда из-за избытка потоков необходимо переключать вычислительные устройства (ядра/процессоры) на выполнение разных потоков, потоки должны восстанавливать состояние кэш-памяти при каждом своем новом возобновлении (что и приводит к появлению дополнительных задержек вычислений); аналогичная проблема возникает и при использовании виртуальной памяти (часть данных приостановленных потоков может быть вытеснена в медленную внешнюю память)
4) Значительная проблема при большом количестве потоков состоит в существенном повышении затрат на организацию синхронизации и взаимоисключения потоков

18. Каким образом минимизация взаимодействия потоков может повысить эффективность выполнения параллельных программ?

1) Безусловно, максимальная эффективность параллельности достигается при полной независимости параллельно выполняемых потоков (при условии равного распределения вычислительной нагрузки). Любое взаимодействие и, как результат, синхронизация деятельности потоков приводит к появлению задержек и снижения быстродействия вычислений.
2) Полностью избежать взаимодействия потоков невозможно (параллельные потоки занимаются решением единой задачи)
3) Нужно уменьшить, по мере возможности, количество необходимых взаимодействий потоков – так, например, если решается задача суммирование значений некоторого числового набора данных, то вместо использования единственной общей переменной для накопления суммы можно сначала вычислить частные суммы для каждого потока в отдельности (без каких-либо синхронизаций), а только потом собрать эти частные суммы вместе
4) Нужно повысить эффективность алгоритмов, выполняемых в критических секциях потоков, и, тем самым, сократить время блокировки общих ресурсов – так, например, если в критической секции необходимо упорядочить данные, то, конечно же, необходимо использовать быстрые алгоритмы сортировки
5) Разделить общие ресурсы для разбиения единственной критической секции на множество раздельных и более редко используемых подсекций – так, при использовании таблицы данных можно организовать блокировку для каждой строки таблицы в отдельности
6) Обеспечить более быстрое выполнение потоков, которые находятся в своих критических секциях – для этого, в частности, можно запретить приостановку таких потоков или повысить их приоритет; применение таких методов обеспечивается обычно средами выполнения параллельных программ

19. Каким образом оптимизация работы с кэш-памятью может повысить эффективность выполнения параллельных программ?

1) При работе с матрицами данных более эффективно проводить обработку элементов по строкам, а не по столбцам, если используется язык фортран

2) При работе с матрицами данных более эффективно проводить обработку элементов по столбцам, а не по строкам, если используется язык С

3) Лучше всегда проводить обработку элементов по строкам, а не по столбцам

20. Каким образом использование библиотек параллельных методов может повысить эффективность выполнения параллельных программ?

1) Использование библиотек позволяет существенно снизить затраты на разработку необходимого параллельного программного обеспечения

2) Применение уже имеющихся многопоточных библиотек крайне полезно – как правило, имеющиеся в этих библиотеках реализации параллельных методов является высокоэффективными

3) Использование таких библиотек может быть и обязательным условием для корректности выполнения многопоточных программ – так, например, при построении программы компилятор должен использовать потоко-ориентированные версии служебных программ среды выполнения (необходимость поддержки многопоточности указывается компилятору при помощи соответствующих управляющий ключей)

21. В чем состоят основы технологии OpenMP и в чем состоят основные преимущества технологии OpenMP?

1) Технология OpenMP позволяет в максимальной степени эффективно реализовать возможности многопроцессорных вычислительных систем с общей памятью, обеспечивая использование общих данных для параллельно выполняемых потоков без каких-либо трудоемких межпроцессорных передач сообщений

2) Сложность разработки параллельной программы с использованием технологии OpenMP в значительной степени согласуется со сложностью решаемой задачи – распараллеливание сравнительно простых последовательных программ, как правило, не требует больших усилий (порою достаточно включить в последовательную программу всего лишь несколько директив OpenMP). Это позволяет, в частности, разрабатывать параллельные программы и прикладным разработчикам, не имеющим большого опыта в параллельном программировании

3) Технология OpenMP обеспечивает возможность поэтапной (инкрементной) разработки параллельных программы – директивы OpenMP могут добавляться в последовательную программу постепенно (поэтапно), позволяя уже на ранних этапах разработки получать параллельные программы, готовые к применению; при этом важно отметить, что программный код получаемых последовательного и параллельного вариантов программы является единым и это в значительной степени упрощает проблему сопровождения, развития и совершенствования программ

4) OpenMP позволяет в значительной степени снизить остроту проблемы переносимости параллельных программ между разными компьютерными системами – параллельная программа, разработанная на алгоритмическом языке С или Fortran с использованием технологии OpenMP, как правило, будет работать для разных вычислительных систем с общей памятью

22. Верны ли следующие утверждения о параллельных программах в рамках технологии OpenMP?

1) Под параллельной программой в рамках OpenMP понимается программа, для которой в специально указываемых при помощи директив местах – параллельных фрагментах – исполняемый программный код может быть разделен на несколько раздельных командных потоков (threads). В общем виде программа представляется в виде набора последовательных (однопотоковых) и параллельных (многопотоковых) участков программного кода

2) В параллельной программе в рамках OpenMP разделение вычислений между потоками осуществляется под управлением соответствующих директив OpenMP. Равномерное распределение вычислительной нагрузки – балансировка (load balancing) – имеет принципиальное значение для получения максимально возможного ускорения выполнения параллельной программы

23. Что понимается под понятием потока (thread) в OpenMP?

1) Потоки могут выполняться на разных процессорах (процессорных ядрах) либо могут группироваться для исполнения на одном вычислительном элементе (в этом случае их исполнение осуществляется в режиме разделения времени). В предельном случае для выполнения параллельной программы может использоваться один процессор – как правило, такой способ применяется для начальной проверки правильности параллельной программы

2) Количество потоков определяется в начале выполнения параллельных фрагментов программы и обычно совпадает с количеством имеющихся вычислительных элементов в системе; изменение количества создаваемых потоков может быть выполнено при помощи целого ряда средств OpenMP

3) Использование в технологии OpenMP потоков для организации параллелизма позволяет учесть преимущества многопроцессорных вычислительных систем с распределенной памятью. Потоки одной и той же параллельной программы могут использовать общие данные для параллельно выполняемых потоков с помощью межпроцессорных передач сообщений

4) Все потоки в параллельных фрагментах программы последовательно перенумерованы от 0 до np-1, где np есть общее количество потоков. Номер потока также может быть получен при помощи функции OpenMP

5) Использование в технологии OpenMP потоков для организации параллелизма позволяет учесть преимущества многопроцессорных вычислительных систем с общей памятью. Прежде всего, потоки одной и той же параллельной программы выполняются в общем адресном пространстве, что обеспечивает возможность использования общих данных для параллельно выполняемых потоков без каких-либо трудоемких межпроцессорных передач сообщений

24. В чем состоит назначение директивы parallel?

1) Когда программа достигает директиву parallel, создается набор (team) из N потоков; исходный поток программы является основным потоком этого набора (master thread) и имеет номер 0

2) Программный код блока, следующий за директивой, дублируется или может быть

разделен при помощи директив между потоками для параллельного выполнения

3) В конце программного блока директивы обеспечивается синхронизация потоков – выполняется ожидание окончания вычислений всех потоков; далее все потоки завершаются – дальнейшие вычисления продолжает выполнять только основной поток (в зависимости от среды реализации OpenMP потоки могут не завершаться, а приостанавливаться до начала следующего параллельного фрагмента – такой подход позволяет снизить затраты на создание и удаление потоков)

25. В чем состоит понятие фрагмента, области и секции параллельной программы?

1) Параллельный фрагмент (parallel construct) – блок программы, управляемый директивой parallel; именно параллельные фрагменты, совместно с параллельными областями, представляют параллельно-выполняемую часть программы; в предшествующих стандартах для обозначения данного понятия использовался термин лексический контекст (lexical extent) директивы parallel
2) Параллельная область (parallel region) – параллельно выполняемые участки программного кода, динамически-возникающие в результате вызова функций из параллельных фрагментов
3) Параллельная секция (parallel section) – часть параллельного фрагмента, выделяемая для параллельного выполнения при помощи директивы section

26. Как осуществляется распараллеливание циклов в OpenMP? Какие условия должны выполняться, чтобы циклы могли быть распараллелены?

1) Для распараллеливания циклов в OpenMP применяется директива for: #pragma omp for [<параметр> ...] <цикл_for>. После этой директивы итерации цикла распределяются между потоками и, как результат, могут быть выполнены параллельно
2) Такое распараллеливание возможно только, если между итерациями цикла есть информационная зависимость

3) Такое распараллеливание возможно только, если между итерациями цикла нет информационной зависимости

27. Как определяются общие и локальные переменные потоков?

1) Параметр shared определяет переменные, которые будут общими для всех потоков
2) Параметр private указывает переменные, для которых в каждом потоке будут созданы локальные копии – они будут доступны только внутри каждого потока в отдельности (значения локальных переменных потока недоступны для других потоков)
3) По умолчанию все переменные программы являются локальными
4) По умолчанию все переменные программы являются общими

28. Что понимается под операцией редукции?

1) Задание коллективной операции, которое происходит при помощи параметра reduction директивы for: reduction (operator: list) где список list задает набор локальных переменных (повторное описание в параметре private переменных из списка list не требуется), для которых должна быть выполнена коллективная операция, а поле operator указывает тип этой коллективной операции. Допустимыми значениями для поля operator являются следующие операции (которые могут быть перегружены): +, -, *, &, |, ^, &&, ||
2) Задание коллективной операции, которое происходит при помощи директивы reduction reduction (operator: list) где список list задает набор локальных переменных (повторное

описание в параметре private переменных из списка list не требуется), для которых должна быть выполнена коллективная операция, а поле operator указывает тип этой коллективной операции. Допустимыми значениями для поля operator являются следующие операции (которые не могут быть перегружены): +, -, *, &, |, ^, &&, ||

29. Какие способы организации взаимоисключения могут быть использованы в OpenMP?

1) Взаимоисключение может быть организовано при помощи неделимых (atomic) операций
2) Взаимоисключение может быть организовано при помощи механизма критических секций (critical sections)
3) Взаимоисключение может быть организовано при помощи специального типа семафоров – замков (locks)

30. Что означает понятие MPI.

1) Стандарт, которому должны удовлетворять средства организации передачи данных
2) Программные средства, которые обеспечивают возможность передачи сообщений и при этом соответствуют всем требованиям стандарта MPI

31. Что понимается в MPI под коммуникатором.

1) Специально созданный служебный объект, объединяющий в своем составе группу процессов
2) Аппаратное устройство, которое используется для организации информационного взаимодействия между процессорами

32. Возможно ли в MPI создание параллельных программ в стиле MIMD, что позволяет создавать процессы с различными исходными текстами.

1) Нет не возможно
2) Возможно, но на практике программисты используют SPMD-модель, в рамках которой для всех процессов используется один и тот же код

33. В чем различие коллективных и парных операций передачи данных.

1) В парных операциях участвуют два процесса программы
2) В парных операциях участвуют два процесса коммуникатора
3) В коллективных операциях участвуют все процессы коммуникатора
4) В коллективных операциях участвуют все процессы программы

34. В чем разница между блокирующими и неблокирующими обменами в MPI.

1) Блокирующие обмены приостанавливают выполнение процессов до завершения работы вызванных функций
2) Неблокирующие обмены осуществляют возврат из функций, не ожидая завершения работы вызванных функций, и таким образом позволяют совмещать процессы передачи сообщений и вычисления

35. В чем состоит понятие тупика. При использовании блокирующих или неблокирующих функций передачи и приема он возможен.

1) Ситуация, когда процессы не могут продолжить работу, потому, что ждут некоторых действий друг от друга. Возможна при использовании неблокирующих функций
2) Ситуация, когда процессы не могут продолжить работу, потому, что ждут некоторых действий друг от друга. Возможна при использовании блокирующих функций

36. Что понимается под операцией MPI_Reduce в MPI.

1) Эта операция определяет коллективную операцию, которая собирает данные от всех процессов и передает результат операции одному процессу
2) Эта операция определяет коллективную операцию, которая собирает данные от некоторых процессов и передает результат операции группе процессов

ДОПОЛНЕНИЯ И ИЗМЕНЕНИЯ В РАБОЧЕЙ ПРОГРАММЕ

за _____/_____ учебный год

В рабочую программу _____
(наименование дисциплины)

Для специальности (тей) _____
(номер специальности)

Вносятся следующие дополнения и изменения:

Дополнения и изменения внес _____
(должность, ФИО, подпись)

Рабочая программа пересмотрена и одобрена на заседании Ученого совета ИПМИ КарНЦ РАН

«___» _____ 20___ г.

Председатель Ученого совета _____ _____
(подпись) (ФИО)