Высокопроизводительные вычислительные системы

Практические работы 1-3. Описание

Цель работ: получить навык распараллеливания вычислительных алгоритмов для исполнения на высокопроизводительных компьютерных системах различных архитектур.

Задачи: для каждого из трёх изучаемых подходов к распараллеливанию (модель обмена сообщениями с применением MPI, модель общей памяти с применением OpenMP, графический вычислитель) реализовать параллельные версии указанных алгоритмов, после чего измерить их быстродействие на сгенерированных тестовых данных.

Порядок выполнения работ:

1. По варианту задания определить номера реализуемых алгоритмов из таблицы 1.

Номер варианта	Алгоритм для MPI	Алгоритм для OpenMP	Алгоритм для GPU
1	1	2	3
2	2	3	5
3	3	5	6
3	5	6	1
4	2	1	3
5	4	2	5
6	7	3	6
7	5	4	2
8	1	7	3
9	2	6	4
10	6	2	7
11	4	1	5
12	6	3	2
13	5	2	1
14	2	5	4
15	1	7	3

2. По таблице 2 определить реализуемые алгоритмы для каждого варианта распараллеливания.

Номер алгоритма	Алгоритм
1	Быстрое преобразование Фурье по алгоритму Кули-Тьюки
2	Ортогонализация матрицы, образованной системой векторов, по методу Грама-Шмидта
3	Прямое преобразование Радона
4	Обратное преобразование Радона
5	LU-разложение матрицы
6	Двумерная свёртка матриц
7	QR-разложение матрицы

3. Для сдачи каждой из работ требуется:

- реализовать соответствующий алгоритм в выбранной модели;
- проверить корректность его работы на тестовых данных;
- измерить время обработки тестовых данных при различном числе обрабатывающих процессов / потоков (минимум от 1 до 8; при усложнении запуска для произвольного

числа параллельных процессов --- для 1,2,4,8); построить график зависимости коэффициента ускорения от числа параллельных процессов / потоков. Необходимо подобрать размеры входных данных таким образом, чтобы время работы последовательной версии программы составило как минимум 1 с.

Контрольные вопросы:

1. Как можно оценить теоретически достижимый коэффициент ускорения каждого из алгоритмов при наличии сведений о быстродействии каждой из используемых арифметических операций и операций доступа к памяти?
2. Как можно выгодно использовать кэширование оперативной памяти в каждом из реализованных алгоритмов? Можно ли теоретически оценить, насколько велик получаемый выигрыш?