Московский Государственный Университет им. М.В. Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Суперкомпьютеров и Квантовой Информатики

Курс: Практикум
Параллельная программа на OpenMP, которая реализует однокубитное квантовое преобразование.

Работу выполнил
Тимачев А.А.
323 группа

Москва 2019

Задание:

1. Реализовать параллельную программу на C++ с использованием OpenMP, которая выполняет однокубитное квантовое преобразование над вектором состояний длины 2^n, где n – количество кубитов, по указанному номеру кубита k. Для работы с комплексными числами использовать стандартную библиотеку шаблонов.

2. Определить максимальное количество кубитов, для которых возможна работа программы на системе Polus.

3. Протестировать программу на системе Polus. В качестве теста использовать преобразование Адамара по номеру кубита:

a) который соответствует номеру в списке группы;

b) 1;

c) n.

Начальное состояние вектора должно генерироваться случайным образом. Заполнить таблицу и построить график зависимости ускорения параллельной программы от числа процессоров для каждого из случаев a)-c)

Результат:

Количество кубитов	Количество нитей	Время работы программы			Ускорение		
		k = 1	k = 7	k = n	k = 1	k = 7	k = n
20	1	0.11223	0.11183	0.10758	1.00000	1.00000	1.00000
	2	0.05428	0.07145	0.07476	2.06772	1.56521	1.43891
	4	0.03548	0.05026	0.03132	3.16351	2.22523	3.43463
	8	0.02165	0.04219	0.02107	5.18299	2.65077	5.10636
24	1	1.59200	1.61495	1.63611	1.00000	1.00000	1.00000
	2	0.92390	1.38526	1.38094	1.72313	1.16580	1.18478
	4	0.61357	0.72478	0.73659	2.59464	2.22817	2.22120
	8	0.46493	0.31046	0.55733	3.42421	5.20173	2.93562
28	1	25.22386	26.43414	25.52509	1.00000	1.00000	1.00000
	2	18.94753	14.85746	13.69106	1.33125	1.77918	1.86436
	4	9.15873	10.06486	8.20922	2.75408	2.62638	3.10932
	8	5.10610	5.52808	4.98575	4.93994	4.78179	5.11960
32	1	479.83545	419.67364	426.90682	1.00000	1.00000	1.00000
	2	261.24335	239.00795	232.23634	1.83674	1.75590	1.83824
	4	163.29141	149.64870	127.15598	2.93852	2.80439	3.35735
	8	117.60566	110.63231	108.36325	4.08004	3.79341	3.93959

Вектор максимальной длины был взят с размером 2^32, так как 256Гбайт = 2 вектора по 2^n элементов размера 16 байт. Отсюда получаем n = 33, но из-за накладных расходов пришлось взять n = 32.

При увеличении количества потоков программа, действительно, начинает работать быстрее, однако эффективность этого ускорения не очень велика. Например, при 8 потоках программа в среднем ускорялась в 4 раза, что говорит об эффективности примерно равной 0,5. Таким образом, можно сделать вывод о том, что программа в целом распараллеливается, но начинает использовать вычислительные ресурсы менее эффективно.