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

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

Работу выполнил
Тимачев А.А.
323 группа

Москва 2019

Задание:

1. Реализовать параллельную программу на C++ с использованием MPI, которая выполняет однокубитное квантовое преобразование над вектором состояний длины 2^n, где n – количество кубитов, по указанному номеру кубита k.

2. Разработать схему распределенного хранения данных на кластерной системе.

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)

Формат командной строки:

./main <n или имя входного файла> <k> [имя файла для вывода]

Структура бинарного файла:

Первые 8 байт(unsigned long long): длина вектора(m).

Последующие m * 16 байт: m комплексных чисел, первые 8 байт(double) — вещественная часть, вторые 8 байт(double) — мнимая часть.

Результат:

Количество кубитов	Количество процессов	Время работы программы			Ускорение		
		k = 1	k = 7	k = n	k = 1	k = 7	k = n
20	1	0.00508	0.00762	0.00726	1.00000	1.00000	1.00000
	2	0.00294	0.00385	0.00375	1.72746	1.98077	1.93365
	4	0.00184	0.00193	0.00209	2.75774	3.94821	3.47225
	8	0.00126	0.00105	0.00102	4.04542	7.26095	7.11471
24	1	0.08614	0.08043	0.07553	1.00000	1.00000	1.00000
	2	0.04539	0.04040	0.04018	1.89790	1.99104	1.87984
	4	0.02226	0.02029	0.01993	3.86946	3.96461	3.78938
	8	0.01357	0.01141	0.01042	6.34908	7.04661	7.25204
28	1	1.21256	1.25479	1.22133	1.00000	1.00000	1.00000
	2	0.70036	0.65362	0.63484	1.73132	1.91974	1.92384
	4	0.35395	0.32785	0.31716	3.42574	3.82733	3.85079
	8	0.22574	0.17439	0.16962	5.37150	7.19525	7.20044
32	1	20.41519	21.35301	20.92405	1.00000	1.00000	1.00000
	2	11.57646	11.35396	11.31618	1.76351	1.88067	1.84904
	4	5.63967	5.33889	5.24933	3.61993	3.99952	3.98604
	8	3.49949	2.95099	2.88726	5.83377	7.23588	7.24702

Как показывают исследования, при увеличении количества процессов программа, действительно, ускоряется. Однако стоит отметить, что в случаях обмена данными между процессами наблюдается более слабое ускорение, в отличии от ситуаций, когда обмен данными не требуется. Таким образом, можно сделать вывод о том, что программа в целом распараллеливается.