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

Практикум на ЭВМ

Отчёт № 2
Параллельная реализация однокубитного преобразования вектора состояния.

Работу выполнил
Малмыгин Г. А.

Москва 2021

Малмыгин Глеб Антонович 323 группа

В прошлом задании мы вычислили, что максимальное число кубитов возможное для единовременного использования на Polus равно 30. Реализовано однокубитное преобразование с помощью MPI, полученные результаты приведены в таблицах ниже.

Описание алгоритма:

В программе есть два варианта ввода начального вектора состояний из файла либо его генерация. В первом случае каждый процесс считывает определенную часть файла и в массив размер которого равен длине вектора состояний поделенному на общее количество процессов, во втором случае каждый процесс генерирует свою часть массива вектора состояний. Для реализации параллельности используется функция MPI_Sendrecv. В каждом процессе имеется два массива длина каждого из которых равна длине вектора состояний деленному на количество процессов. Второй вектор необходим, чтобы получить необходимые элементы начального вектора состояний для вычисления конечного вектора. Элементы необходимые для вычисления результата текущий процесс получает со входа и с помощью функций передачи получает необходимые данные для вычисления вектора. Каждый процесс вычисляет номер процесса, от которого ему необходимо получить данные и, если вычисленный номер не совпадает с текущим процессом происходит обмен данными. Вычисление номера процесса, от которого будет происходить получение данных зависит от номеров индексов элементов в общем векторе и номеров, полученных их инверсией в бите с номером k.

Тестирование программы:

Тестирование производилось на локальной машине на количестве процессов 1, 2, 4, 8, количество кубитов 16. Тестирование происходит следующим образом, при запуске на одном процессе происходит генерация входного вектора, вычисляется вектор результат, оба вектора записываются в разные файлы. Далее при количестве процессов более одного, программа считывает входной вектор из файла и генерирует результат, далее сравнивает полученный результат с вектором из выходного файла, который был получен на одном процессе.

Вычисление результатов:

Для замера времени выполнения алгоритма используется функция MPI_Wtime. Все вычисления произведены на Polus.

Таблица 1 Результаты преобразования Адамара номер кубита k = 1

Количество кубитов	Количество процессоров	Время работы программы в сек	Ускорение
20	1	0.0909868	1
	2	0.0458473	1.984561795
	4	0.0229579	3.963202209
	8	0.0119102	7.639401521
	16	0.0114679	7.934041978
	32	0.00863023	10.54280129
	64	0.00427139	21.30144988
24	1	1.45571	1
	2	0.752448	1.934632028
	4	0.367129	3.965118528
	8	0.189842	7.668008133
	16	0.0971575	14.98299153
	32	0.0681102	21.37286339
	64	0.0343773	42.345094
28	1	23.5355	1
	2	11.6453	2.021029943
	4	6.37218	3.693476958
	8	3.02219	7.787564647
	16	1.54548	15.22860212
	32	0.980373	24.00667909
	64	0.466446	50.45707327

Максимально возможное число кубитов (30)	1		
	2		
	4	23.5635	1
	8	11.8964	1.980725261
	16	6.14488	4.160130059
	32	3.315	7.108144796
	64	1.89652	12.42459874

Таблица 2 Результаты преобразования Адамара номер кубита k = 11

Количество кубитов	Количество процессоров	Время работы программы в сек	Ускорение
20	1	0.0909739	1
	2	0.0455327	1.997990455
	4	0.0229512	3.9637971
	8	0.0115699	7.862980665
	16	0.00846789	10.74339652
	32	0.00769615	11.82070256
	64	0.00429893	21.16198682
24	1	1.458	1
	2	0.738039	1.975505359
	4	0.370341	3.936912197
	8	0.186012	7.838203987
	16	0.116364	12.52964834
	32	0.0494588	29.47908158
	64	0.0342297	42.59458891
28	1	23.3342	1
	2	11.7403	1.987530131
	4	5.82549	4.005534298
	8	2.96873	7.859994004
	16	1.55238	15.03124235
	32	0.928583	25.12882532
	64	0.472283	49.407241
	1		
	2		

Максимально возможное число кубитов (30)	4	24.2689	1
	8	11.7304	2.068889381
	16	6.19338	3.918522681
	32	3.47618	6.981485424
	64	1.76765	13.72947133

Таблица 3 Результаты преобразования Адамара номер кубита k = n

Количество кубитов	Количество процессоров	Время работы программы в сек	Ускорение
20	1	0.0908856	1
	2	0.0473659	1.918798123
	4	0.0234135	3.881760523
	8	0.0157496	5.770660842
	16	0.00829738	10.95352991
	32	0.00805434	11.28405307
	64	0.00494385	18.383567746
24	1	1.50461	1
	2	0.77084	1.951909605
	4	0.375588	4.006011907
	8	0.213794	7.037662423
	16	0.12612	11.92998731
	32	0.0712384	21.12077194
	64	0.039303	38.28231942
28	1	23.3167	1
	2	11.9109	1.957593465
	4	5.95698	3.914181347
	8	3.05866	7.623174854
	16	1.60346	14.54149152
	32	1.03373	22.55588984
	64	0.625265	37.29090865
	1		
	2		

Максимально возможное число кубитов (30)	4	23.7813	1
	8	12.1317	1.960261134
	16	6.40367	3.713698551
	32	3.93035	6.050682509
	64	2.1707	10.95559036

Графики зависимости ускорения от количества процессов, N = 20, 28, 30

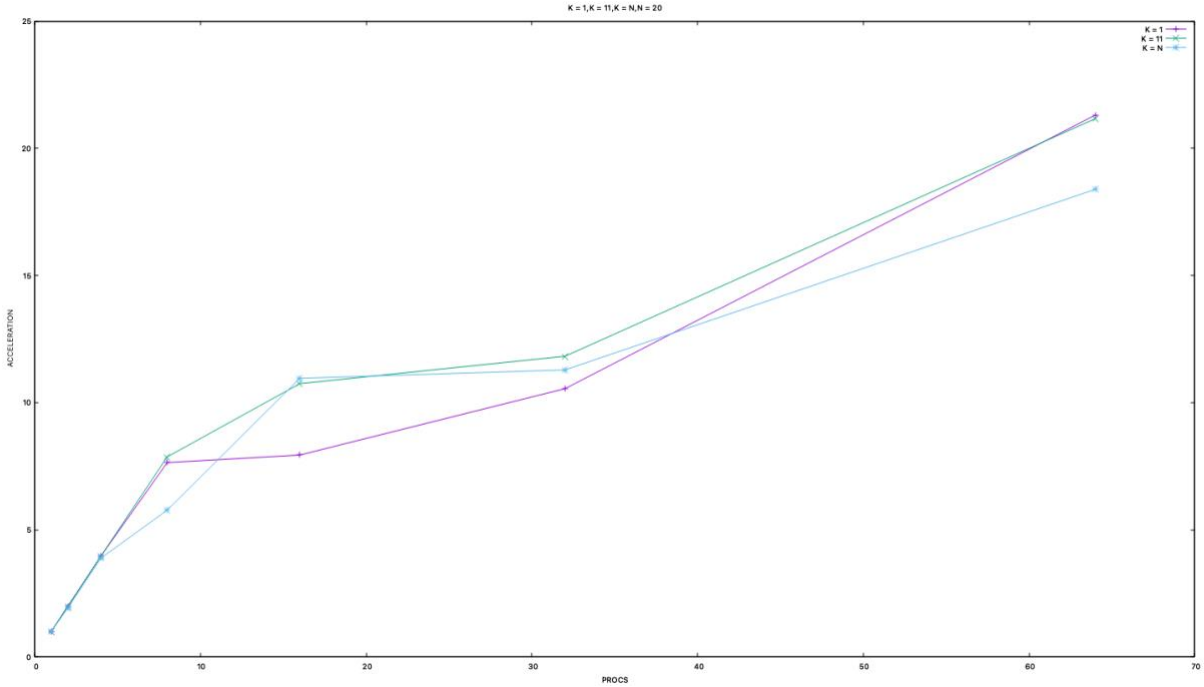

График 1 Количество кубитов равно 20, K = 1, K = 11, K = N

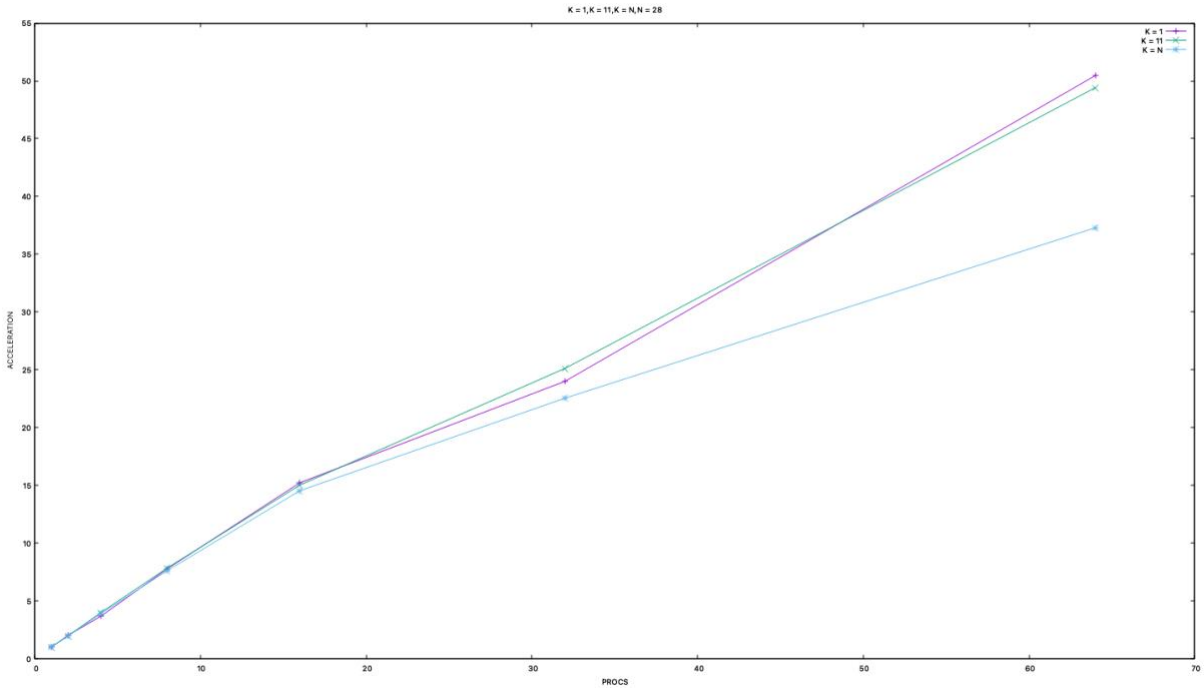

График 2 Количество кубитов равно 28, K = 1, K = 11, K = N

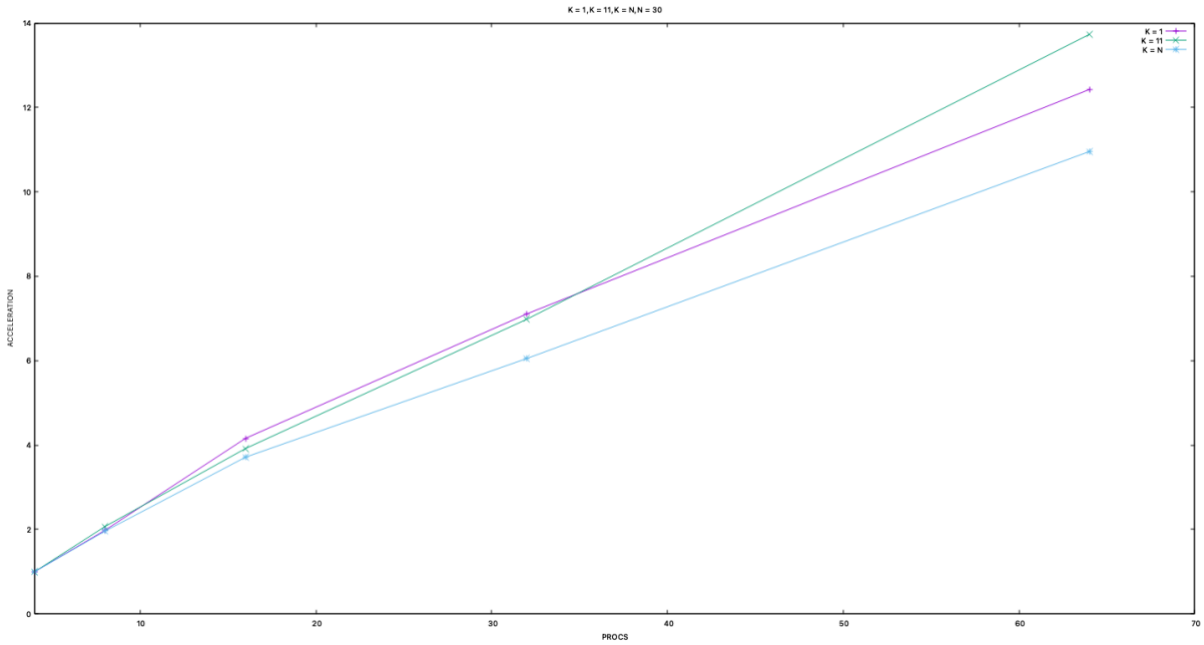

График 3 Количество кубитов равно 30, K = 1, K = 11, K = N, ускорение относительно четырех процессов

Падение коэффициента ускорения с ростом количества процессов связано прежде всего с количеством обменов, которые происходят между процессами, что и замедляет работу параллельной программы при большом количестве процессов.

Время при 30 кубитах на 1 и 2 процессах вычислить не удалось, программа работает слишком долго, поэтому ускорение при таком количестве процессов было вычислено относительно 4 процессов.

Низкое ускорение на 20 кубитах может быть связано с зашумлением на Polus.