Московский государственный университет имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра суперкомпьютеров и квантовой информатики

Отчёт по практикуму на ЭВМ за 2 семестр 4 курса

Выполнил:

Козлов М.В.

423 группа

Москва, 2018

Постановка задачи и алгоритм

Реализовать параллельную программу, которая решает систему линейных алгебраических уравнений (Ax = B) методом отражений.

В начале работы происходят простейшие проверки на командную строку. Затем идет расчёт размера блока матрицы, который будет находится на одном узле, и его генерация. В данной программе матрица генерируется по формуле 1.0 / ((i + 1.0) + (j + 1.0) + 1.0) (диагональному элементу добавляется размерность матрицы), а каждый элемент вектора B (правая часть) задаётся как сумма соответствующей строки матрицы A. Распределение по узлам – циклическое.

Далее следует алгоритм приведения матрицы к верхнетреугольному виду методом отражений. Он состоит из главного цикла, который проходится по всем столбцам матрицы кроме последнего. По номеру столбца все процессы понимают, какой узел должен его обрабатывать: соответствующий узел строит по текущему столбцу вектор $x^{(i)}$, совпадающий с текущем вектором во всех координатах кроме первой, а первая заменяется на $(a_{kk}^{(k-1)} - ||a_1^{(k-1)}||)$, где вектор *a* – текущий вектор. Далее вектор $x^{(i)}$ нормируется и рассылается всем процессам, которые на лету умножают свои вектора на матрицу отражения, которая построена по вектору $x^{(i)}$ (U($x^{(i)}$) = I - $2x^{(i)}x^{(i)*}$). Так же они умножают на матрицу *U* вектор B (в каждом процессе они одинаковые). На каждом шаге на диагонали появляется норма вектора, а координаты ниже диагонали зануляются, и размерность вектора $x^{(i)}$ уменьшается на 1. После завершения цикла мы получаем распределённую по узлам верхнетреугольную матрицу.

По полученной матрице мы вычисляем решение параллельным методом Гаусса. Как и в методе отражений, цикл проходит по всем столбцам, но уже не с начала, а с конца. Соответствующей текущему столбцу узел вычисляет корень с тем же номером и рассылает всем данный столбец в последней координате которой лежит вычисленный корень. Каждый получивший узел умножает полученный столбец на полученный корень и вычитает из своего вектора B. В итоге мы получаем вектор решений на каждом узле.

После получения решения алгоритм вычисляет невязку ||Ax – B||, где x – вектор решений. Каждый узел умножает покоординатно свои вектора матрицы

A на вектор решений, после чего также покоординатно складывает все вектора и отправляет на узел с рангом 0. Получив вектора, 0 узел складывает их все покоординатно и получает вектор Ax, по которому и считается невязка ||Ax – B||.
Далее следует вывод на экран: количество узлов, размерность матрицы, время каждого этапа, невязка, а также вектор решений.

Для удобства построения таблиц и графиков был написан маленький скрип на питоне, который из полученных файлов *.out переписывает информацию о данных для таблиц в отдельные файлы, то есть убирает вектор решений из файла.

Компиляция и запуск

В Makefile команда make компилирует программу reflection.cpp в файл reflection. Команды make run и make run1 в цикле ставят в очередь на Blue Gene\P программы на 1, 32, 64, 128, 256, 512, 1024 узлах с размерностями матрицы 1024, 2048, 3072. Каждая комбинация количество узлов-размерность матрицы запускается дважды, для уточнения времени. Команда make convert преобразует выходные файлы для анализа.

Таблицы и графики

Количество узлов	Размерность матрицы	Время выполнения(секунды) Отражение, Гаусс, Невязка		
1	1024	12.4366	0.0202409	0.0760638
	2048	101.91	0.0806914	0.582807
	3072	417.222	0.185999	1.33648
2	1024	6.193	0.0401657	0.0148562
	2048	50.8782	0.147242	0.284568
	3072	208.91	0.314879	0.661805
4	1024	3.1728	0.0292692	0.00798018
	2048	25.6521	0.11828	0.0680939
	3072	104.828	0.258713	0.328467
8	1024	1.61882	0.0293544	0.00411726
	2048	12.9859	0.116063	0.0168204
	3072	52.7192	0.258499	0.0679075
16	1024	0.843939	0.0294793	0.0026807
	2048	6.64394	0.116379	0.00946635
	3072	26.7077	0.25901	0.0235403
32	1024	0.456299	0.0294538	0.00259435
	2048	3.45709	0.116302	0.00699816
	3072	13.6654	0.258627	0.013992
64	1024	0.261775	0.0298627	0.00380553
	2048	1.87082	0.117612	0.00845221
	3072	7.16455	0.260402	0.0138586
128	1024	0.163616	0.0302228	0.0071150
	2048	1.07655	0.118612	0.0143529
	3072	3.91275	0.261763	0.0214729
256	1024	0.112524	0.0310126	0.0144831
	2048	0.675249	0.121238	0.0283186
	3072	2.28422	0.264912	0.0401434
512	1024	0.0833878	0.0301999	0.0311677
	2048	0.470077	0.119154	0.0577992
	3072	1.46141	0.261996	0.0806981

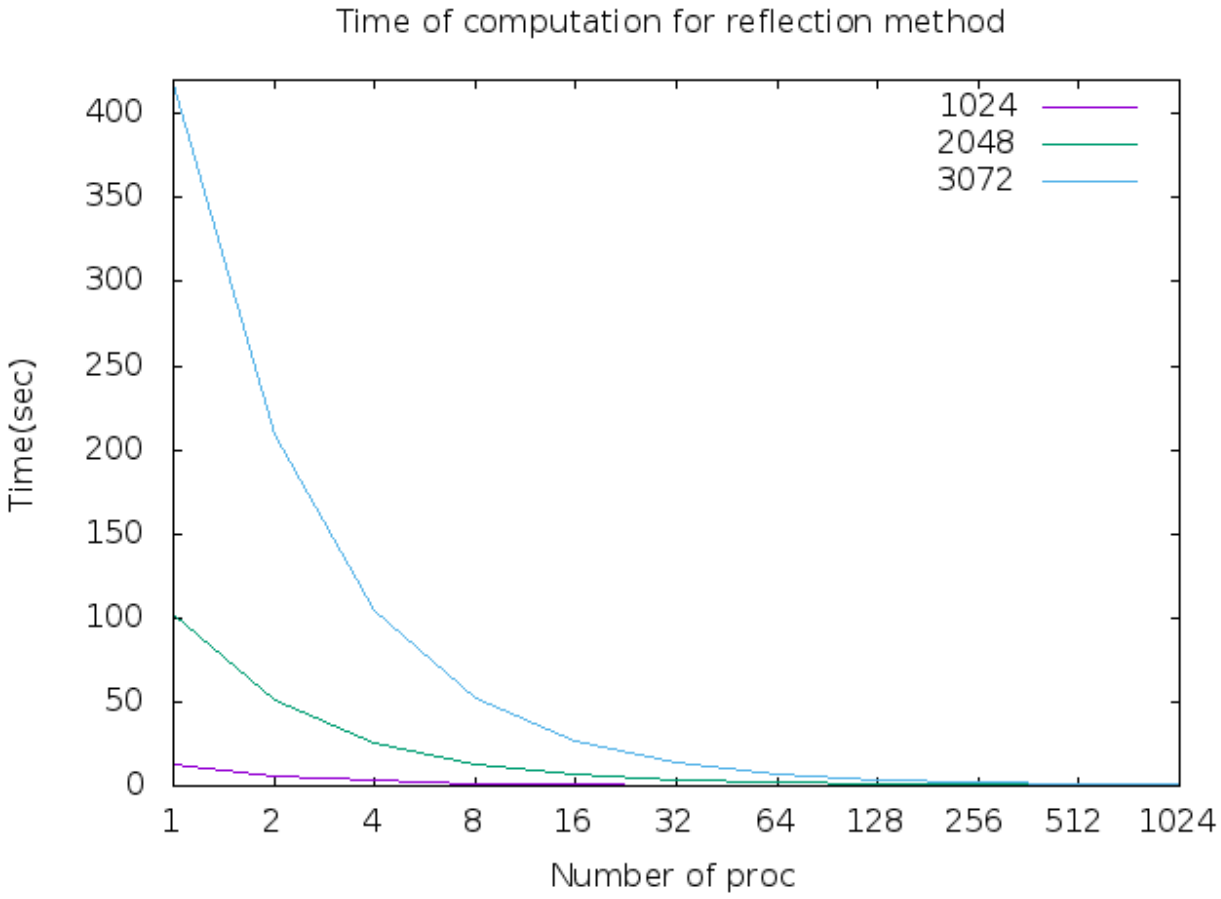
1024	1024	0.0727728	0.0296113	0.0738306
	2048	0.372409	0.120992	0.125433
	3072	1.05566	0.264476	0.174445

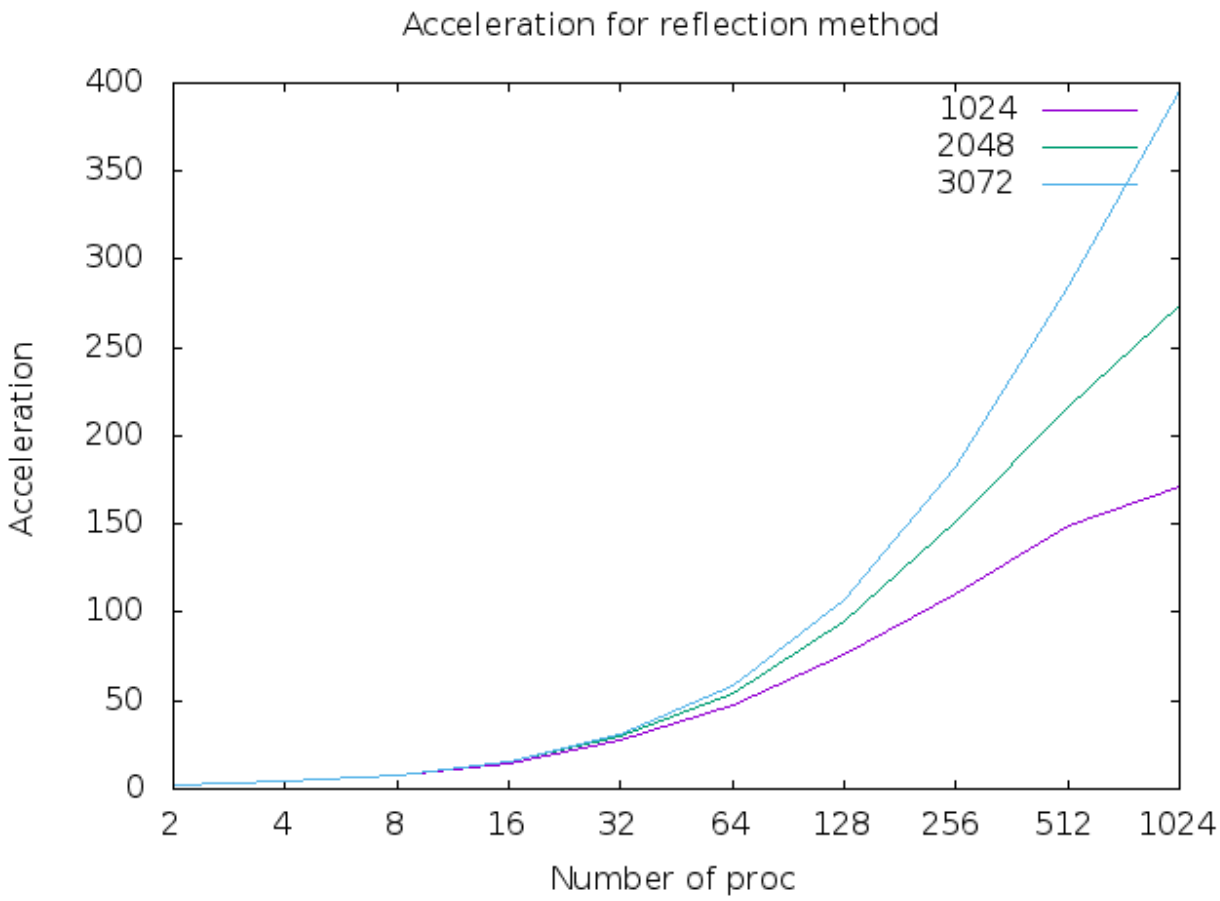
Количество узлов	Размерность матрицы	Ускорение = T1 / Tp Отражение, Гаусс, Невязка		
2	1024	2,0081705	0,5039349	5,1200038
	2048	2,0030190	0,5480189	2,0480412
	3072	1,9971375	0,5906999	2,0194468
4	1024	3,9197554	0,6915426	9,5315895
	2048	3,9727742	0,6822066	8,5588724
	3072	3,9800626	0,7189395	4,0688410
8	1024	7,6825095	0,6895355	18,4743737
	2048	7,8477426	0,6952379	34,6488193
	3072	7,9140427	0,7195347	19,6808894
16	1024	14,7363731	0,6866140	28,3746036
	2048	15,3387899	0,6933502	61,5661792
	3072	15,6217870	0,7181151	56,7741278
32	1024	27,25537422	0,687208442	29,31902018
	2048	29,47854988	0,693809221	83,28003361
	3072	30,53126875	0,719178585	95,51743854
64	1024	47,50873842	0,677798726	19,98770211
	2048	54,47343945	0,686081352	68,95320869
	3072	58,23422267	0,714276388	96,43686953
128	1024	76,01090358	0,669722858	10,69062544
	2048	94,66350843	0,680297103	40,60552223
	3072	106,6313974	0,710562608	62,24031221
256	1024	110,5239771	0,652666981	5,251900491
	2048	150,9221043	0,665561953	20,58036061
	3072	182,6540351	0,702116174	33,29264586
512	1024	149,141721	0,670230696	2,440468819
	2048	216,7942699	0,677202612	10,08330565

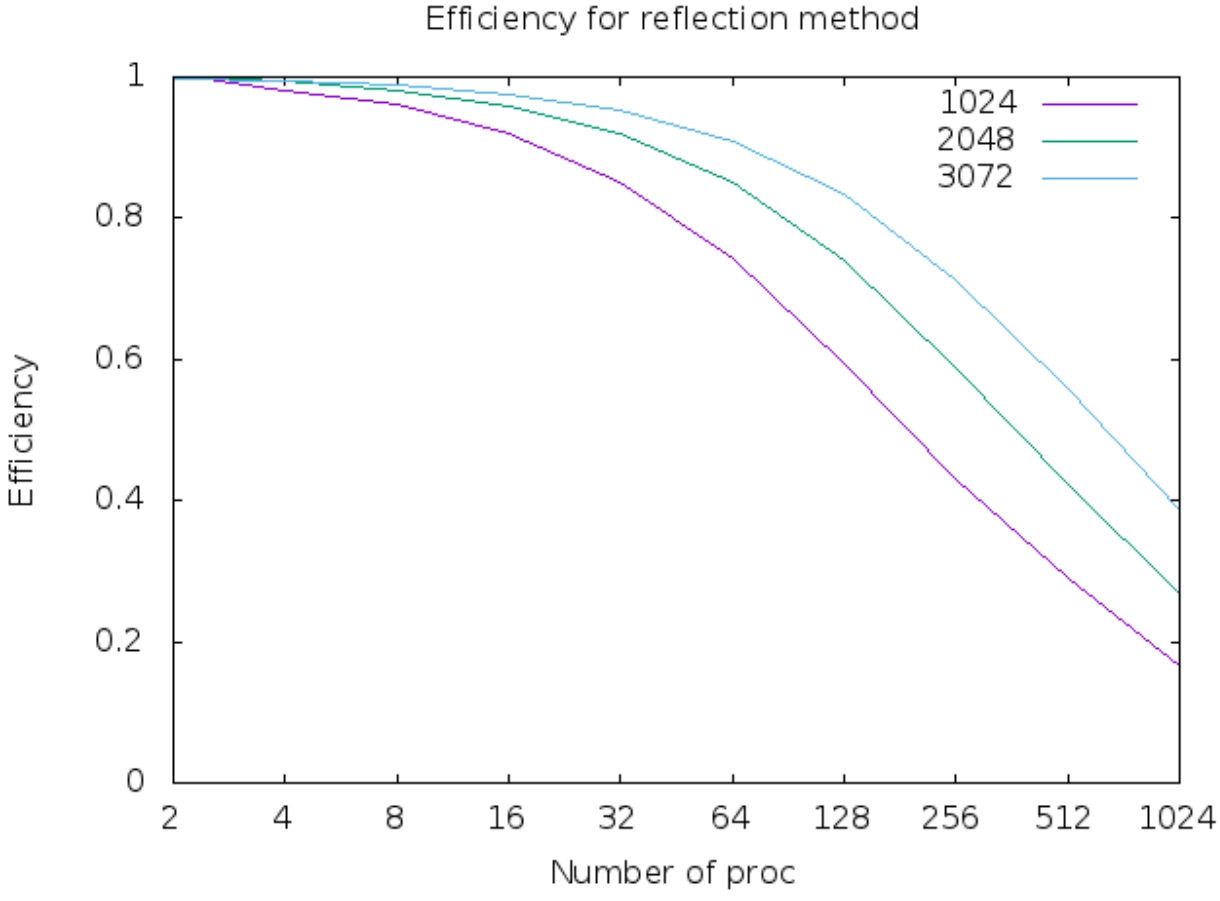
	3072	285,4927775	0,709930686	16,56148038
1024	1024	170,8962689	0,683553238	1,030247621
	2048	273,6507442	0,666915168	4,646361005
	3072	395,223841	0,703273643	7,661325919

Количество узлов	Размерность матрицы	Эффективность = S / p Отражение, Гаусс, Невязка		
2	1024	1,0040853	0,2519675	2,5600019
	2048	1,0015095	0,2740095	1,0240206
	3072	0,9985688	0,2953500	1,0097234
4	1024	0,9799389	0,1728857	2,3828974
	2048	0,9931936	0,1705517	2,1397181
	3072	0,9950157	0,1797349	1,0172103
8	1024	0,9603137	0,0861919	2,3092967
	2048	0,9809678	0,0869047	4,3311024
	3072	0,9892553	0,0899418	2,4601112
16	1024	0,9210233	0,0429134	1,7734127
	2048	0,9586744	0,0433344	3,8478862
	3072	0,9763617	0,0448822	3,5483830
32	1024	0,851730444	0,021475264	0,916219381
	2048	0,921204684	0,021681538	2,60250105
	3072	0,954102148	0,022474331	2,984919954
64	1024	0,742324038	0,010590605	0,312307845
	2048	0,851147491	0,010720021	1,077393886
	3072	0,909909729	0,011160569	1,506826086
128	1024	0,593835184	0,00523221	0,083520511
	2048	0,73955866	0,005314821	0,317230642
	3072	0,833057792	0,00555127	0,486252439
256	1024	0,431734286	0,00254948	0,020515236
	2048	0,58953947	0,002599851	0,080392034
	3072	0,713492325	0,002742641	0,130049398
512	1024	0,291292424	0,001309044	0,004766541
	2048	0,423426308	0,001322661	0,019693956

	3072	0,557603081	0,001386583	0,032346641
1024	1024	0,166890888	0,000667532	0,001006101
	2048	0,267237055	0,000651284	0,004537462
	3072	0,385960782	0,000686791	0,007481764




Acceleration for reflection method
Acceleration
Number of proc
1024
2048
3072
0
50
100
150
200
250
300
350
400
2
4
8
16
32
64
128
256
512
1024


Efficiency for reflection method
Efficiency
Number of proc
1024
2048
3072
0
0.2
0.4
0.6
0.8
1
2
4
8
16
32
64
128
256
512
1024