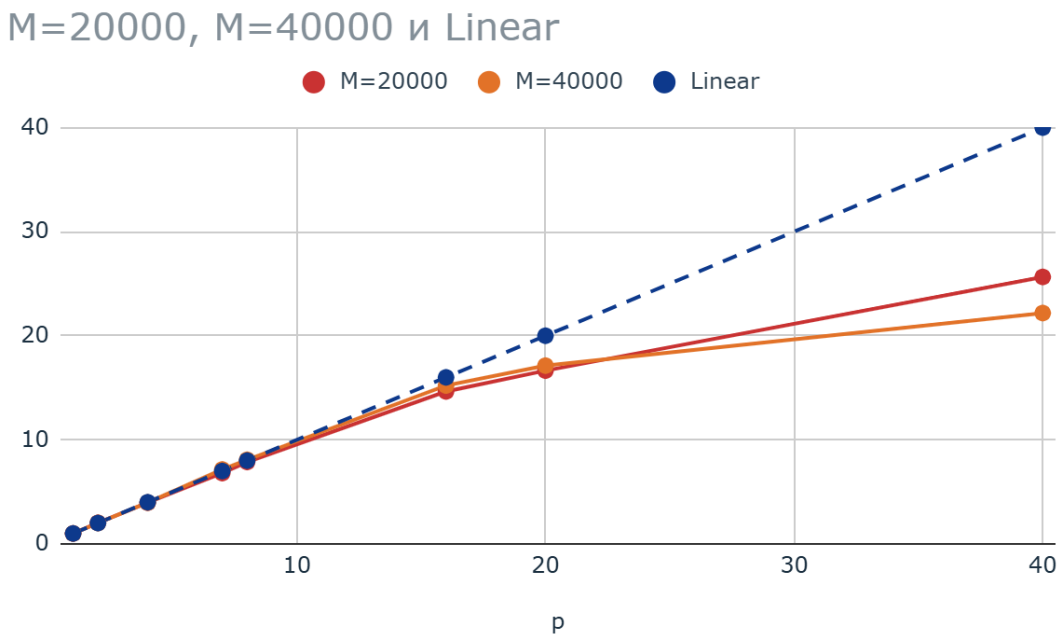
Задание 1

Умножение матрицы на вектор с параллельной инициализацией данных

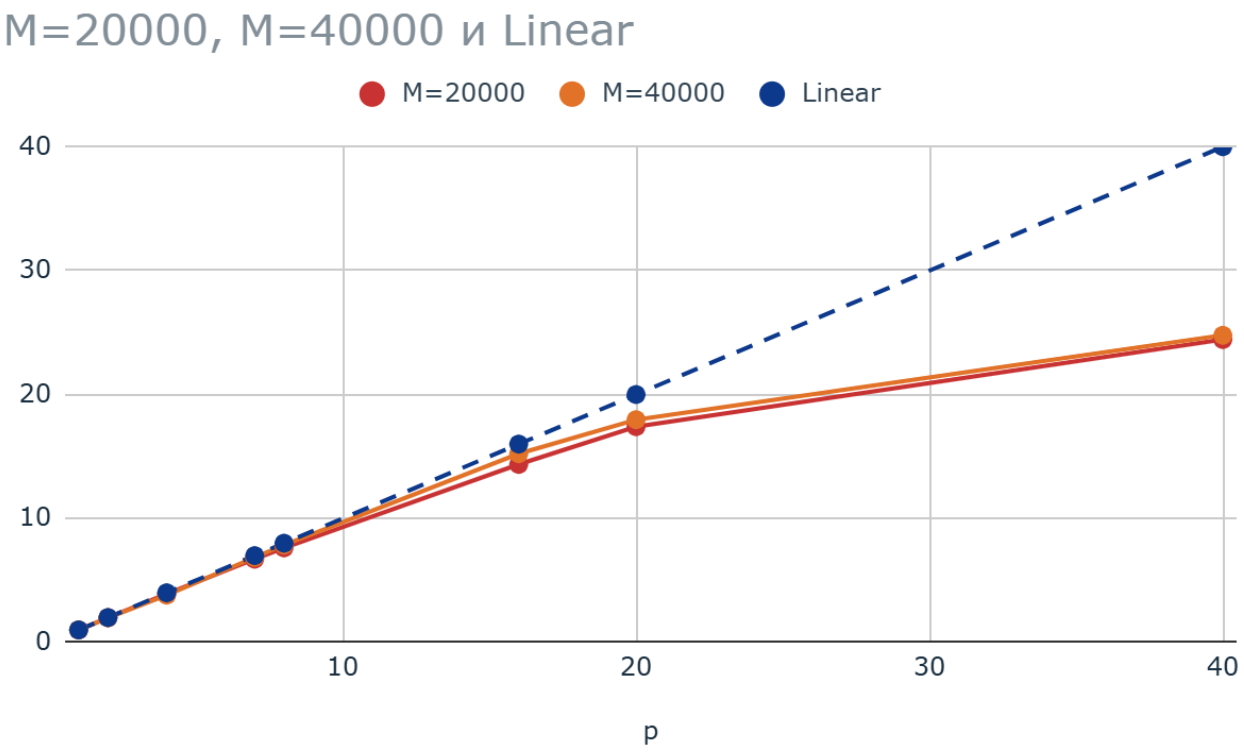
Контейнер 1 (vector)

	Количество потоков															
	1		2		4		7		8		16		20		40	
M=N	T(1)	S(1)	T(2)	S(2)	T(4)	S(4)	T(7)	S(7)	T(8)	S(8)	T(16)	S(16)	T(20)	S(20)	T(40)	S(40)
20000	1333,10	1	659,15	2,02	336,89	3,96	195,99	6,80	169,37	7,87	91,02	14,65	80,07	16,65	51,97	25,65
40000	5659,50	1	2819,19	2,01	1425,58	3,97	791,72	7,15	700,55	8,08	371,99	15,21	330,46	17,13	255,15	22,18



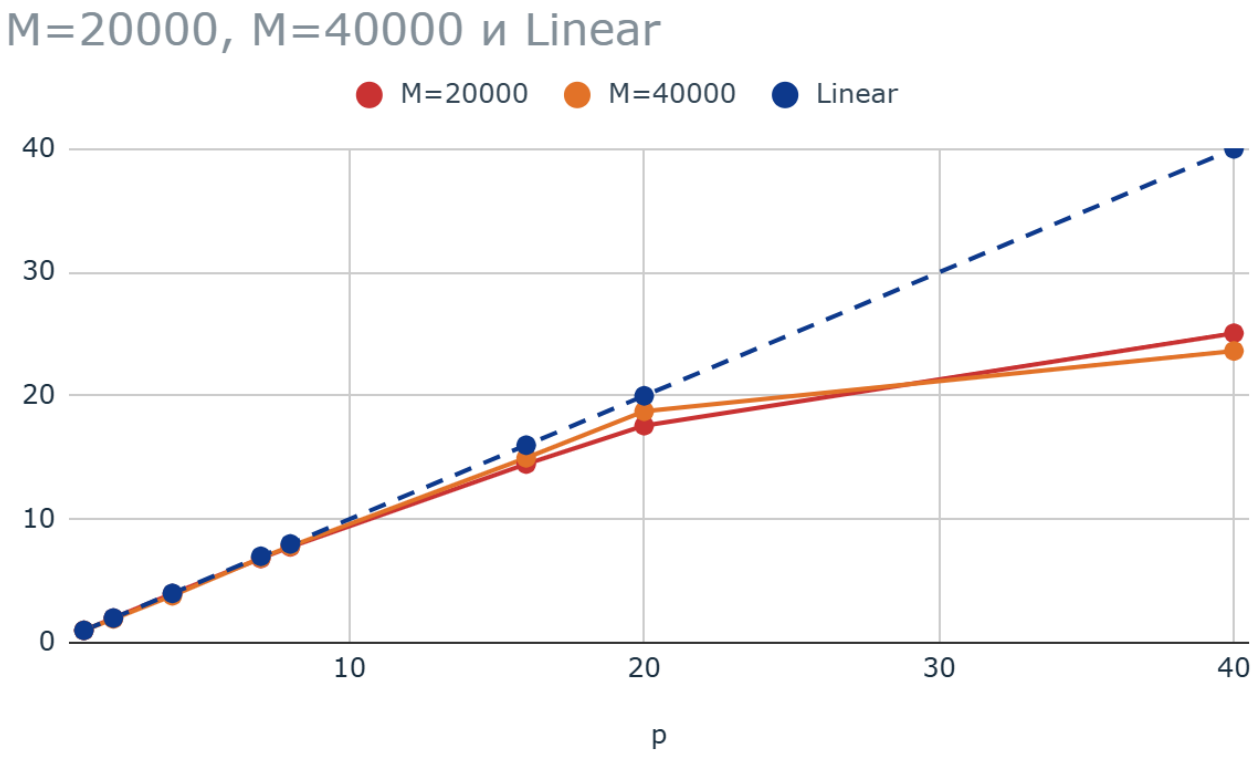
Контейнер 2 (array)

M=N	Количество потоков															
	1		2		4		7		8		16		20		40	
	T(1)	S(1)	T(2)	S(2)	T(4)	S(4)	T(7)	S(7)	T(8)	S(8)	T(16)	S(16)	T(20)	S(20)	T(40)	S(40)
20000	1310,83	1	659,29	1,99	335,99	3,90	195,26	6,71	171,91	7,63	91,27	14,36	75,31	17,41	53,64	24,44
40000	5397,62	1	2723,64	1,98	1416,46	3,81	786,81	6,86	687,70	7,85	354,59	15,22	300,36	17,97	217,84	24,78



Контейнер 3 (deque)

	Количество потоков															
	1		2		4		7		8		16		20		40	
M=N	T(1)	S(1)	T(2)	S(2)	T(4)	S(4)	T(7)	S(7)	T(8)	S(8)	T(16)	S(16)	T(20)	S(20)	T(40)	S(40)
20000	1331,38	1	669,30	1,99	335,60	3,97	195,12	6,82	171,45	7,77	91,98	14,47	75,71	17,58	53,11	25,07
40000	5393,37	1	2786,19	1,94	1418,83	3,80	785,60	6,87	690,57	7,81	360,10	14,98	287,67	18,75	228,21	23,63



При увеличении количества потоков ускорение растёт почти линейно до 8 потоков. Далее наблюдается снижение эффективности масштабирования, особенно на 40 потоках, что обусловлено ограничениями на использование памяти, пропускной способностью и накладными расходами на управление потоками.

Все 3 типа контейнеров показали примерно одинаковые результаты, но вариант 1 дает наилучшее ускорение, а вариант 3 чуть лучше по времени на больших массивах.

Для практического использования выбрать вариант 1 (std::vector). Он показывает лучшую масштабируемость и остаётся стабильно производительным на всех тестируемых конфигурациях.