Задание 1 – Умножения матрицы на вектор с параллельной инициализацией массивов

M = N	Количество потоков															
	1		2		4		7		8		16		20		40	
	T1	**S1**	**T2**	**S2**	**T4**	**S4**	**T7**	**S7**	**T8**	**S8**	**T16**	**S16**	**T20**	**S20**	**T40**	**S40**
20000 (3052 MiB)	*0,59*	*1,05*	*0,31*	*1,94*	*0,17*	*3,58*	*0,10*	*6,10*	*0,09*	*6,94*	*0,05*	*12,61*	*0,04*	*15,63*	*0,03*	*22,54*
40000 (12207 MiB)	*2,58*	*0,90*	*1,30*	*1,79*	*0,67*	*3,44*	*0,38*	*6,15*	*0,32*	*7,32*	*0,17*	*13,90*	*0,13*	*17,36*	*0,07*	*32,48*

Анализируя коэффициент ускорения Sp(n) для различных размеров матриц и числа потоков, можно сделать следующие выводы о масштабируемости программы:

1. Рост ускорения при увеличении числа потоков

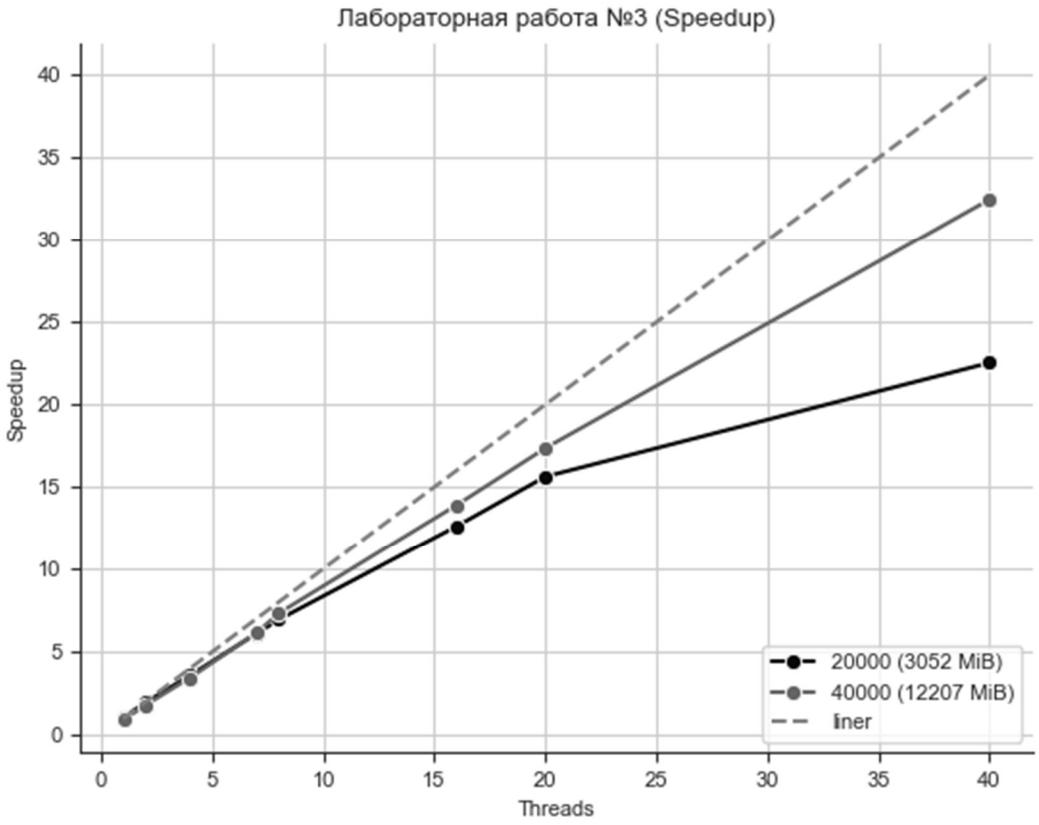
Ускорение растет с увеличением количества потоков, но не линейно. Это типично для параллельных программ, так как накладные расходы на управление потоками и доступ к памяти могут снижать эффективность.

2. Уменьшение эффективности при большом числе потоков

При 40 потоках ускорение (например, Sp(40)=22.54 для 20000×20000 и Sp(40)=32.48 для 40000×40000) значительно меньше, чем теоретически ожидаемые 40 раз. Это указывает на накладные расходы и возможные узкие места в системе (например, ограничение пропускной способности памяти).

3. Нелинейность ускорения

Для малых чисел потоков (1, 2, 4) ускорение близко к линейному, но при дальнейшем росте начинает снижаться. Например, при 8 потоках Sp(8)=6.94 для 20000×20000, что довольно близко к идеальному Sp(8)=8. Однако при 16 потоках ускорение уже составляет Sp(16)=12.61, что указывает на возрастающие накладные расходы.



Вывод о масштабируемости (В сравнении с OpenMP версией):

OpenMP показывает лучшее ускорение, особенно в диапазоне 8–20 потоков.

Версия на OpenMP более эффективно использует многопоточность и ресурсы системы, особенно при умеренном количестве потоков, а для версии на основе jthread возможно требуется дополнительная доработка кода чтобы добиться более высокой эффективности.