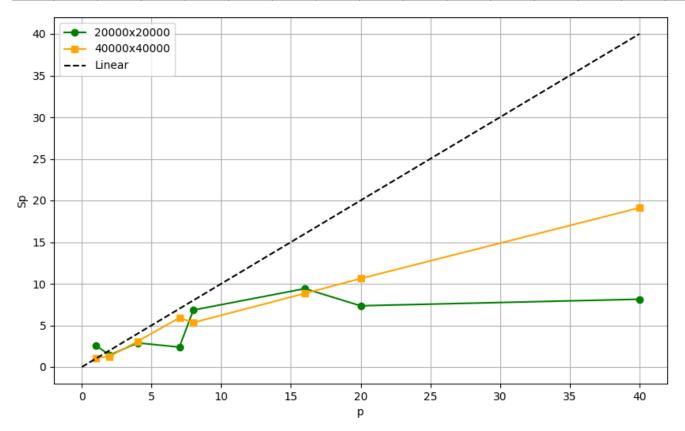
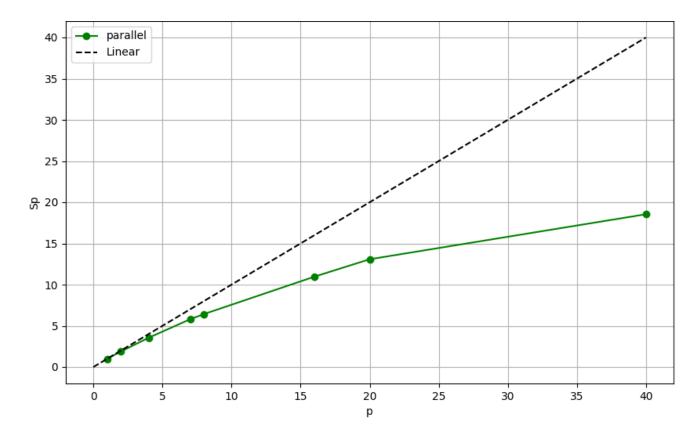
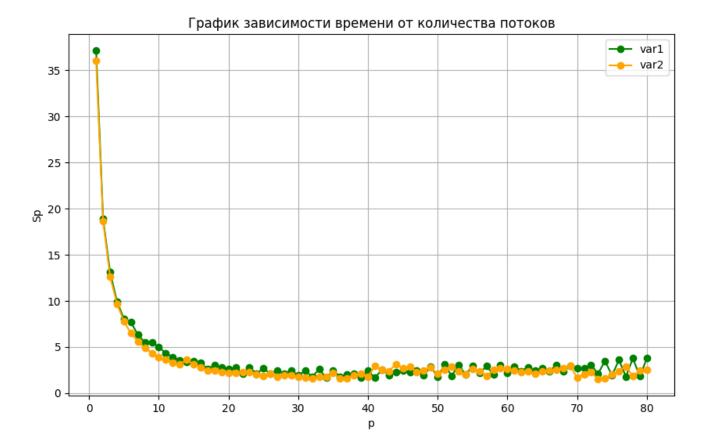
M=N	Количество потоков														
	2			4		7		8		16		20		40	
	T1	T2	S2	T4	S4	T7	S 7	Т8	S8	T16	S16	T20	S20	T40	S40
20 000	1.14635	0.807729	1.43107	0.520701	2.89398	0.497372	2.39194	0.176223	6.83853	0.147684	9.41256	0.182012	7.35171	0.138861	8.1387
40 000	4.39538	4.34346	1.28147	1.84001	3.07253	0.734257	5.91159	0.853263	5.33379	0.520285	8.83889	0.44865	10.62144	0.263271	19.1142



Результаты показывают, что использование OpenMP для параллельного умножения матрицы на вектор ускоряет выполнение по сравнению с последовательным вариантом. При увеличении числа потоков время выполнения уменьшается, но с ростом числа потоков оно уменьшается незначительно.



Результаты показывают, что использование OpenMP для параллельного численного интегрирования ускоряет выполнение по сравнению с последовательным вариантом. При увеличении числа потоков время выполнения уменьшается, но с ростом числа потоков оно уменьшается незначительно.





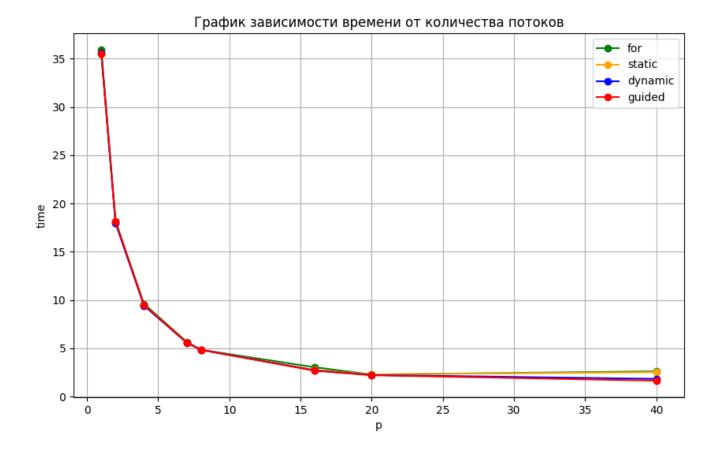
Входные данные:

t = 0.00001

e = 1e-6

N=M=4000

По графикам видно, что второй вариант программы выполняет большинство тестов быстрее. Из-за чего целесообразно использовать второй вариант.



Если приглядеться, можно заметить, что guided – лучший вариант для использования в #pragma omp for schedule().

Краткая характеристика:

Architecture: x86_64

CPU op-mode(s): 32-bit, 64-bit

Address sizes: 46 bits physical, 48 bits virtual

Byte Order: Little Endian

CPU(s): 80
On-line CPU(s) list: 0-79
Vendor ID: GenuineIntel

Model name: Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz

CPU family: 6
Model: 85
Thread(s) per core: 2
Core(s) per socket: 20
Socket(s): 2
Stepping: 7

CPU max MHz: 3900.0000 CPU min MHz: 1000.0000 BogoMIPS: 5000.00

Caches (sum of all):

L1d: 1.3 MiB (40 instances)
L1i: 1.3 MiB (40 instances)
L2: 40 MiB (40 instances)
L3: 55 MiB (2 instances)

NUMA:

NUMA node(s): 2 node 0 size: 385636 MB node 1 size: 387008 MB

Операционная система:

PRETTY_NAME="Ubuntu 22.04.5 LTS"

NAME="Ubuntu" VERSION_ID="22.04"