CPU info:

```
Architecture:            x86_64
CPU(s):                  80
Vendor ID:               GenuineIntel
Model name:              Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz
Thread(s) per core:      2
Core(s) per socket:      20
Socket(s):               2
CPU max MHz:             3900.0000
CPU min MHz:             1000.0000
```

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)
```

Наименование сервера:

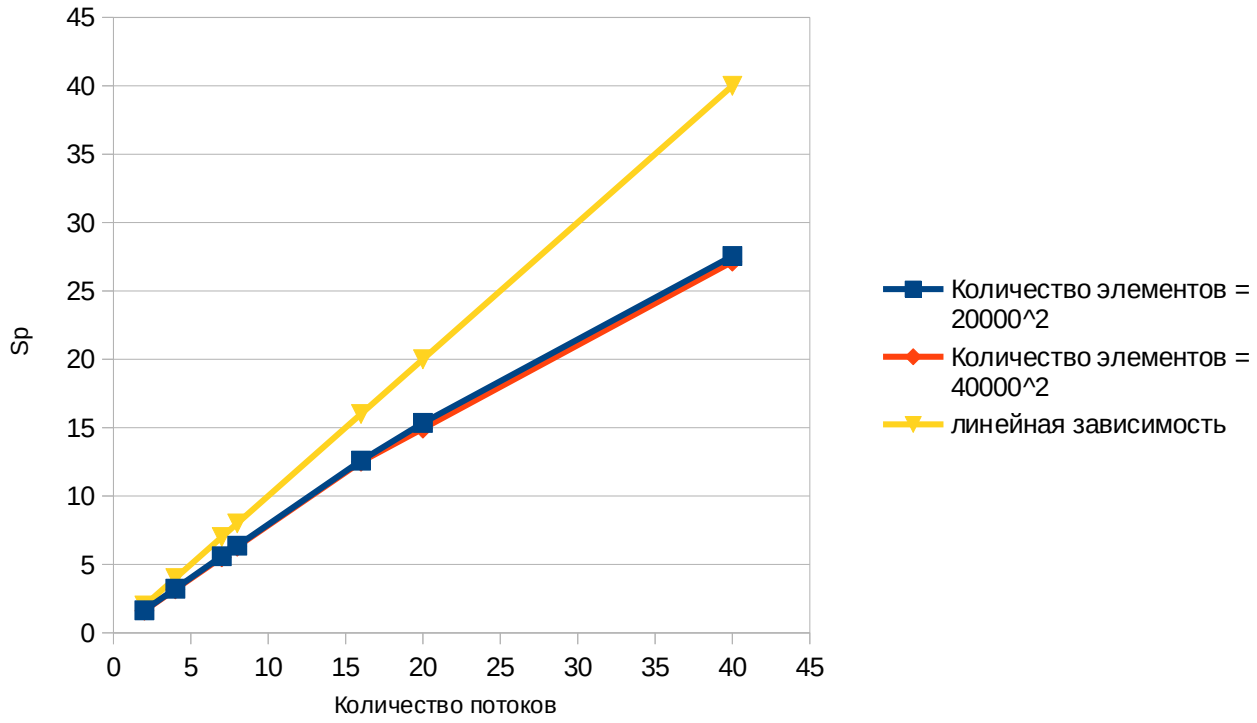
ProLiant XL270d Gen10

NUMA node:

```
2 nodes
node 0 size: 385636 MB
node 1 size: 387008 MB
```

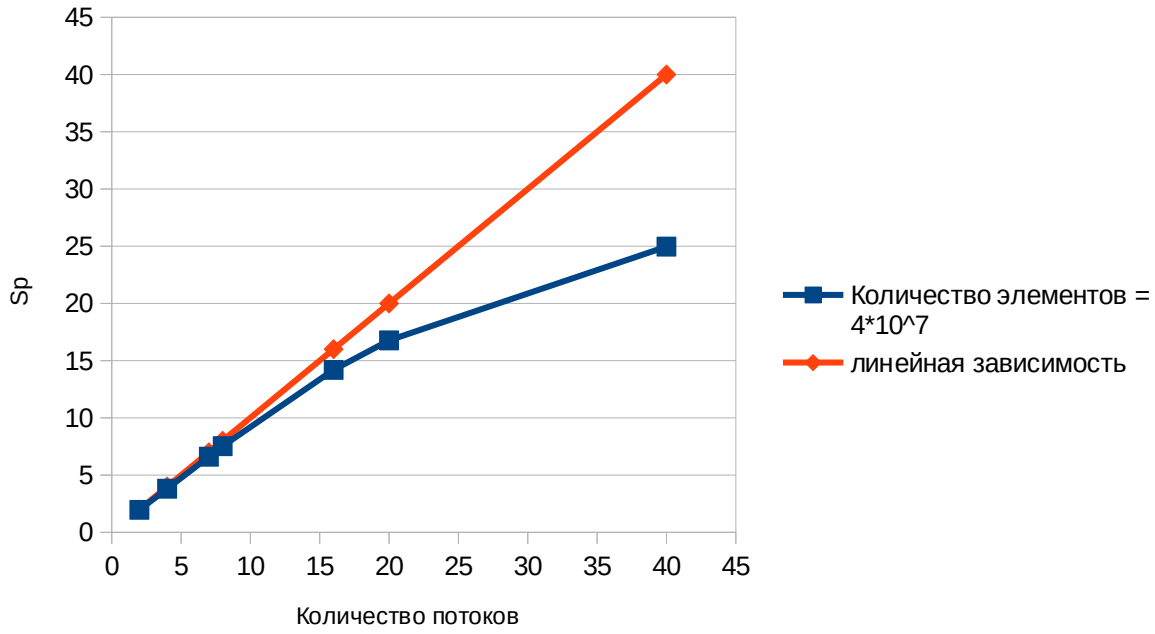
OS: Ubuntu 22.04.5 LTS

M=N	Количество потоков														
	2			4		7		8		16		20		40	
	T1	T2	S2	T4	S4	T7	S7	T8	S8	T16	S16	T20	S20	T40	S40
20000	1,060	0,645	1,643	0,328	3,224	0,189	5,604	0,166	6,371	0,084	12,591	0,069	15,355	0,038	27,55
40000	4,190	2,589	1,618	1,320	3,173	0,760	5,512	0,665	6,301	0,335	12,504	0,280	14,926	0,154	27,14



Распараллеливание приносит стабильный окололинейный результат в обоих случаях. Соответственно, данную параллельную программу можно адекватно масштабировать и она будет показывать хорошие показатели производительности относительно последовательной программы.

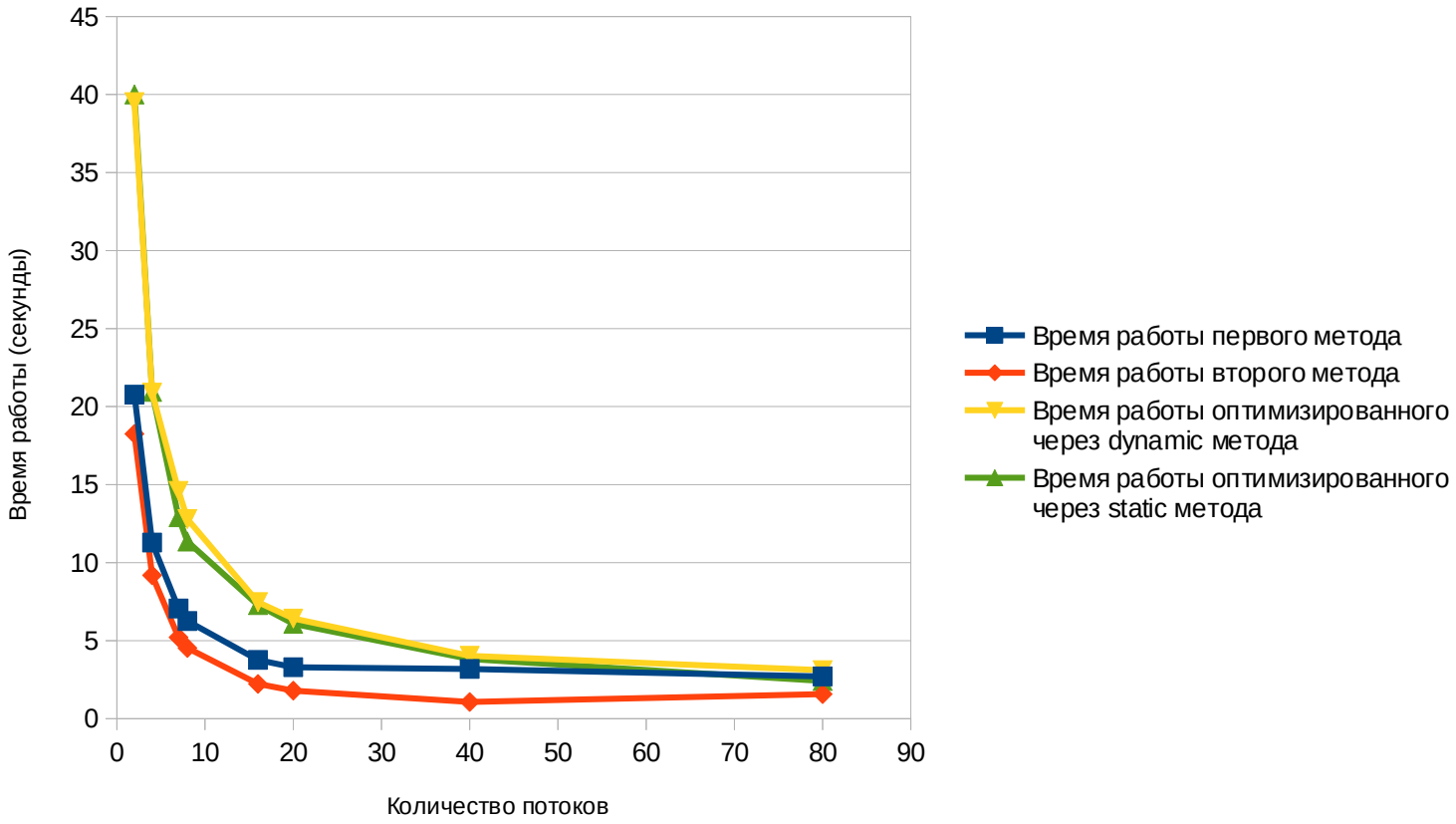
steps	Количество потоков														
	2			4		7		8		16		20		40	
	T1	T2	S2	T4	S4	T7	S7	T8	S8	T16	S16	T20	S20	T40	S40
4 * 10^7	0,467	0,237	1,968	0,122	3,813	0,070	6,605	0,062	7,530	0,032	14,188	0,027	16,771	0,018	24,96

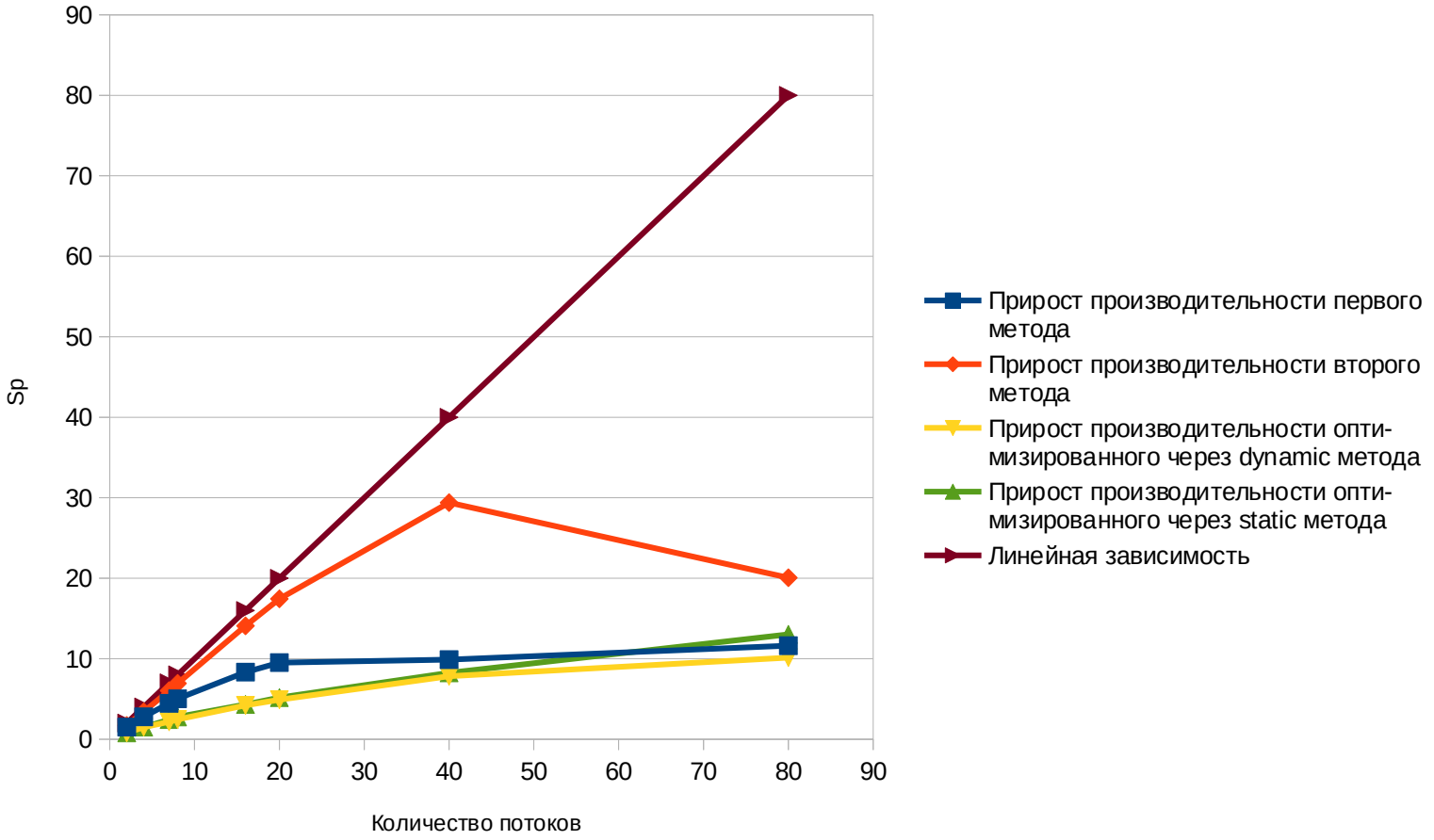


Распараллеливание приносит стабильный окололинейный результат до 20 потоков включительно. При 40 потоках прирост не настолько сильный, но он все равно ощутим.

Соответственно, параллельную программу можно адекватно масштабировать и она будет показывать хорошие показатели производительности относительно последовательной программы.

Ver.	Количество потоков															
	2			4		7		8		16		20		40		80	
	T1	T2	S2	T4	S4	T7	S7	T8	S8	T16	S16	T20	S20	T40	S40	T80	S80
1	31,54	20,77	1,51	11,29	2,79	7,06	4,46	6,26	5,03	3,77	8,34	3,30	9,52	3,187	9,89	2,71	11,63
2	31,54	18,25	1,72	9,19	3,42	5,21	6,04	4,53	6,94	2,23	14,09	1,80	17,45	1,07	29,40	1,57	20,07
dynamic	31,54	39,51	0,79	20,89	1,50	14,60	2,16	12,79	2,46	7,47	4,21	6,42	4,91	4,03	7,82	3,11	10,14
static	31,54	40,00	0,78	20,94	1,50	12,91	2,44	11,35	2,77	7,28	4,32	6,08	5,17	3,82	8,25	2,42	13,03





Итак, самый эффективный алгоритм — второй(вся программа в одном #pragma omp parallel). В его случае распараллеливание больше чем на 40 потоков не имеет смысла.
Schedule оптимизировал первый алгоритм только с параметром static на 80 потоках. В остальных случаях schedule только ухудшал производительность программы.

Таким образом, самый эффективный вариант в данном случае — поместить весь цикл в одну #pragma omp parallel.