1. OpenMP warm-up

(a) Each threads take $(n+2) * (n-1) $ ms to execute the parallel region. The waiting time would be $(n - 1) / 4 * 4 $ ms for one thread to waiting for other thread.

(b) Each threads take $(n+1) * (n-1) $ ms.

(c) Each threads take $(n) * (n-1) $ ms.

(D) Yes.With the key word **nowait**. OpenMp can eliminate teh waiting time. Each threads take $(n) * (n-1) $ ms.

2. Parallel Scan in OpenMP

1. Run in CIMS Linux Server. 4 AMD EPYC Processor. 7.6G Memory.
2. GCC 12.2, OpenMP4.0

Thread nums	sequential	1	2	4	8	16
time (s)	0.278617s	0.544569s	0.295552s	0.184230s	0.187318	0.183572s

OpenMP version of 2D Jacobi/Gauss-Seidel smoothing

I fist implement serial 2D Jacobi and GS smoothing. And then implement OpenMP version with Jacobi and color GS.

1. Run in CIMS Linux Server. 4 AMD EPYC Processor. 7.6G Memory.
2. GCC 12.2, OpenMP4.0

N = 100, Iter = 5000

Model Name	Thread Nums	time
Jacobi	1	0.0834974s
Jacobi OMP	1	0.194962s
Jacobi OMP	2	0.145284s

Model Name	Thread Nums	time
Jacobi OMP	4	0.201126s
Jacobi OMP	8	0.826496s
GS	1	0.216483s
Color GS OMP	1	0.241206s
Color GS OMP	2	0.238263s
Color GS OMP	4	0.250183s
Color GS OMP	8	0.537342s

N = 1000, Iter = 5000

Model Name	Thread Nums	time
Jacobi	1	20.6051s
Jacobi OMP	1	23.042492s
Jacobi OMP	2	12.503416s
Jacobi OMP	4	9.049808s
Jacobi OMP	8	14.781276s
GS	1	26.2275s
Color GS OMP	1	23.098190s
Color GS OMP	2	21.470304s
Color GS OMP	4	20.326765s
Color GS OMP	8	23.653441s

N = 10000, Iter = 500

Model Name	Thread Nums	time
Jacobi	1	190.404s
Jacobi OMP	1	217.741425s
Jacobi OMP	2	164.181616s
Jacobi OMP	4	147.963412s
Jacobi OMP	8	141.527666s
Jacobi OMP	16	138.033301s
Jacobi OMP	32	139.466289s

Model Name	Thread Nums	time
GS	1	260.705
Color GS OMP	1	245.863702s
Color GS OMP	2	219.466680s
Color GS OMP	4	200.948334s
Color GS OMP	8	208.010856s