

More work sharing

COMPUTATIONAL MATH, SCIENCE AND ENGINEERING DEPARTMENT

MICHIGAN STATE
UNIVERSITY

OpenMP loop scheduling

```

5312 0000488B 15470B00 004889D6 4889C7E8 6E050000 488B1535 0B000048 89D64889 C7E85C05 0000488D 35790600 00488B05 140B0000 4889C7E8 7C050000 BE080000
5376 004889C7 E85D0500 00488B15 000B0000 4889D648 89C7E827 050000E8 10050000 4889C348 8D354A06 0000488B 05070A00 004889C7 E83F0500 004889D6 4889C7E8
5440 1C050000 488B15C5 0A000048 89D64889 C7E8EC04 0000E8CF 04000048 89C3488D 351E0600 00488B05 9C0A0000 4889C7E8 04050000 4889D648 89C7E8E1 04000048
5504 8B158A0A 00004889 D64889C7 E8B10400 00488B15 780A0000 4889D648 89C7E89F 04000048 8D35E705 0000488B 05570A00 004889C7 E8BF0400 000B0800 00004889
5568 C7E8A004 0000488B 15430A00 004889D6 4889C7E8 6A040000 488B1531 0A000048 89D64889 C7E85804 0000488D 35B70500 00488B05 100A0000 4889C7E8 78040000
5632 BE040000 004889C7 E8590400 00488B15 FC090000 4889D648 89C7E823 040000E8 F4030000 660F7EC3 488D3588 05000048 8B05D209 00004889 C7E83A04 0000660F
5696 6EC34889 C7E8A004 0000488B 15BF0900 004889D6 4889C7E8 E6030000 E8B10300 00660F7E C3488D35 5B050000 488B0595 09000048 89C7E8FD 03000066 0F6EC348
5760 89C7E8CD 03000048 8B158209 00004889 D64889C7 E8A90300 00488D35 36050000 488B0561 09000048 89C7E8C9 030000BE 18000000 4889C7E8 9E030000 488B154D
5824 09000048 89D64889 C7E87403 0000488B 153B0900 004889D6 4889C7E8 62030000 488D3509 05000048 8B051A09 00004889 C7E88203 0000BE08 00000048 89C7E863
5888 03000048 8B150609 00004889 D64889C7 E82D0300 00E8E602 00006648 0F7EC348 C050488B 05D80800 004889C7 E8430300 0066480F 6EC34889 C7E80603
5952 0000488B 15C70800 004889D6 4889C7E8 EE020000 E8A10200 0066480F 7EC3488D 35AC0400 00488B05 9C080000 4889C7E8 04030000 66480F6E C34889C7 E8C70200
6016 00488B15 88080000 4889D648 89C7E8AF 02000048 8D358704 0000488B 05670800 004889C7 E8CF0200 00BE3500 00004889 C7E8A402 0000488B 15530800 004889D6
6080 4889C7E8 7A020000 488B1541 08000048 89D64889 C7E86802 0000488D 355B0400 00488B05 20080000 4889C7E8 88020000 BF100000 004889C7 E8690200 00488B15
6144 0C080000 4889D648 89C7E89F 04000048 8B158209 00004889 D64889C7 E8A90300 00488D35 36050000 488B0561 09000048 89C7E8C9 030000BE 18000000 4889C7E8 9E030000 488B154D
6208 00004889 D64889C7 E8B10400 00488B15 780A0000 4889D648 89C7E89F 04000048 8D35E705 0000488B 05570A00 004889C7 E8BF0400 000B0800 00004889
6272 4889D648 89C7E8CD 03000048 8B158209 00004889 D64889C7 E8A90300 00488D35 36050000 488B0561 09000048 89C7E8C9 030000BE 18000000 4889C7E8 9E030000 488B154D
6336 00004889 D64889C7 E8B10400 00488B15 780A0000 4889D648 89C7E89F 04000048 8D35E705 0000488B 05570A00 004889C7 E8BF0400 000B0800 00004889
6400 07000048 8B05F606 00004889 C7E86C01 0000C9C3 554889E5 BEFFF000 00BF0100 0000E8A5 FFFF5FD5 C3554889 E5B80080 FFFF5DC3 554889E5 B8FF7F00 005DC355
6464 4889E5B8 00000080 5DC35548 89E5B8FF FFFF7F5D C3554889 E5488B00 00000000 0000805D C3554889 E5488BFF FFFF7F5D C3554889 E58B0589 01000066
6528 0F6EC05D C3554889 E58B057D 01000066 0F6EC05D C3554889 E5488B00 00000000 00100066 480F6EC0 5DC35548 89E5488B FFFF7F5D FFFF7F5D 66480F6E C05DC355
6592 4889E548 B8000000 00000000 00B80A10 00004889 45F08955 F8D6D0F0 5DC35548 89E548C7 C0FFFFF7 FFB8FE7F 00004889 45F08955 F8D6D0F0 5DC35548 89E5488B FFFF7F5D
6656 FF252A06 0000FF25 2C060000 FF252E06 0000FF25 30606000 FF253206 0000FF25 34060000 FF253606 0000FF25 38060000 FF253A06 0000FF25 3C060000 FF253E06
6720 0000FF25 40060000 FF254206 0000FF25 44060000 FF254606 0000FF25 48060000 FF254A06 0000FF25 4C060000 FF254E06 0000FF25 50060000 FF255206 0000FF25
6784 54060000 4C8D1D95 05000041 53FF2585 05000090 68000000 00E9E6FF FFFF6819 000000E9 DCF7FFFF 682B0000 00E9D2FF FFFF683D 000000E9 C8FFFFF7 684F0000
6848 00E9BEFF FFFF6861 000000E9 B4FFFFF7 68730000 00E9AAFF FFFF6885 000000E9 A0FFFFF7 68970000 00E996FF FFFF68B7 000000E9 8CFFFFF7 68F70000 00E982FF
6912 FFFF0000 00000000 FFFF77F7 FFFF77FF 53697A65 206F6620 73686F72 743A0053 6D616C6C 65737420 73686F72 743A004C 61726765 73742073 686F7274 3A005369
6976 7A05206F 6620696E 743A0053 6D616C6C 65737420 696E743A 006C6172 70696734 3A005369 66206C6F 66E73A00 66206C6F 66E73A00 66206C6F 66E73A00
7040 673A004C 61726765 7374206C 6F6E673A 0053697A 65206F66 206C6F6E 67206C6F 6E672069 6E743A00 53697A65 206F6620 666C6F61 743A0053 6D616C6C 65737420
7104 666C6F61 743A004C 61726765 73742066 6C6F6174 3A004469 67697473 20696E20 6D617469 7376312C 20666C6F 61743A00 53697A65 206F6620 646F7562 6C653A00
7168 536D616C 6C657374 20646F75 626C653A 004C6172 67657374 20646F75 626C653A 00446967 69747320 696E206D 61746973 73612C20 646F7562 6C653A00 53697A65
7232 206F6620 6C6F6E67 20646F75 626C653A 00536D61 6C6C6573 74206C6F 6E67206A 6F75626C 653A004C 61726765 7374206C 6F6E6720 646F7562 6C653A00 00000000
7296 44696769 74732069 6E206D61 74697373 612C206C 6F6E6720 646F7562 6C653A00 14000000 00000000 017A5200 01781001 100C0708 90010000 34000000 1C000000
7360 69FCFFFF FFFFFFFF 0B000000 00000000 00040100 00000E10 86020403 00000000 06040600 00000C07 08000000 00000000 00000000 54000000 3CFCFFFF FFFFFFFF
7424 0B000000 00000000 00040100 00000E10 86020403 00000000 06040600 00000C07 08000000 00000000 34000000 00000000 00000000 00000000 00000000
7488 00040100 00000E10 86020403 00000000 06040600 00000C07 08000000 00000000 34000000 C4000000 E2FBFFFF FFFFFFFF 0B000000 00000000 00040100 00000E10
7552 86020403 00000000 06040600 00000C07 08000000 00000000 34000000 FC000000 B5FBFFFF FFFFFFFF 10000000 00000000 00040100 00000E10 86020403 00000000
7616 06040B00 00000C07 08000000 00000000 34010000 8DFBFFFF FFFFFFFF 10000000 00000000 00000000 00040100 00000E10 86020403 00000000 06040B00 00000C07
7680 08000000 00000000 34000000 6C010000 65FBFFFF FFFFFFFF 10000000 00000000 00040100 00000E10 86020403 00000000 06040B00 00000C07 08000000 00000000
7744 34000000 A4010000 3DFBFFFF FFFFFFFF 10000000 00000000 00040100 00000E10 86020403 00000000 06040B00 00000C07 08000000 00000000 34000000 DC010000
7808 15FBFFFF FFFFFFFF 15000000 00000000 00040100 00000E10 86020403 00000000 06041000 00000C07 08000000 00000000 00000000 00000000 14020000 F2FAFFFF FFFFFFFF
7872 15000000 00000000 00040100 00000E10 86020403 00000000 06041000 00000C07 08000000 00000000 34000000 C4020000 CFFAFFFF FFFFFFFF 1F000000 00000000
7936 00040100 00000E10 86020403 00000000 06041A00 00000C07 08000000 00000000 34000000 84020000 B6FAFFFF FFFFFFFF 1C000000 00000000 00040100 00000E10
8000 86020403 00000000 06041700 00000C07 08000000 00000000 34000000 BC020000 02FAFFFF FFFFFFFF 6A050000 00000000 00040100 00000E10 86020403 00000000
8064 06040500 00000303 04600500 000C0708 34000000 F4020000 349FFF FF FFFFFFFF 48000000 00040100 00000E10 86020403 00000000 06044300 00000C07
8128 08000000 00000000 34000000 2C030000 44F9FFFF FFFFFFFF 15000000 00000000 00040100 00000E10 86020403 00000000 06041000 00000C07 08000000 00000000
8192 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 A8190000 01000000 95190000 01000000 D8190000 01000000
8256 BF190000 01000000 75190000 01000000 85190000 01000000 4A190000 01000000 3F190000 01000000 65190000 01000000 55190000 01000000 34190000 01000000
8320 29190000 01000000 9A1A0000 01000000 9E1A0000 01000000 A81A0000 01000000 B21A0000 01000000 BC1A0000 01000000 C61A0000 01000000 D81A0000 01000000
8384 DA1A0000 01000000 E41A0000 01000000 EE1A0000 01000000 F81A0000 01000000 14190000 01000000 00000000 00000000 00000000 00000000 00000000
8448 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
8512 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
8576 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

CMSE 822, FS21, W.F. Punch

The default

The default behavior for a for loop is to divide the iterations by the number of threads and assign each thread that number of iterations, in order



What about unequal load?

What happens if the load at each iteration is unequal?

One thread is busy and the others, having finished, just waiting around



nasty progressive function

```
double f (int iters){  
    double result = 0.0;  
  
    for (int j = -iters; j<= iters; ++j)  
        result += sqrt(atan(j));  
  
    return result;  
}
```

Range grows with i. Bigger i, longer time.

atan and sqrt can't really be optimized.




```
double t1 = omp_get_wtime();  
# pragma omp parallel for  
num_threads(t_cnt) reduction(+:  
result)  
    for(int i=0; i<iters; ++i)  
        result += f(i);  
double t2 = omp_get_wtime();
```

Work is unevenly distributed, threads at the end do much more work.



schedule

Can adapt the "schedule" of thread to work in a couple of ways:

- static
- dynamic
- guided
- auto



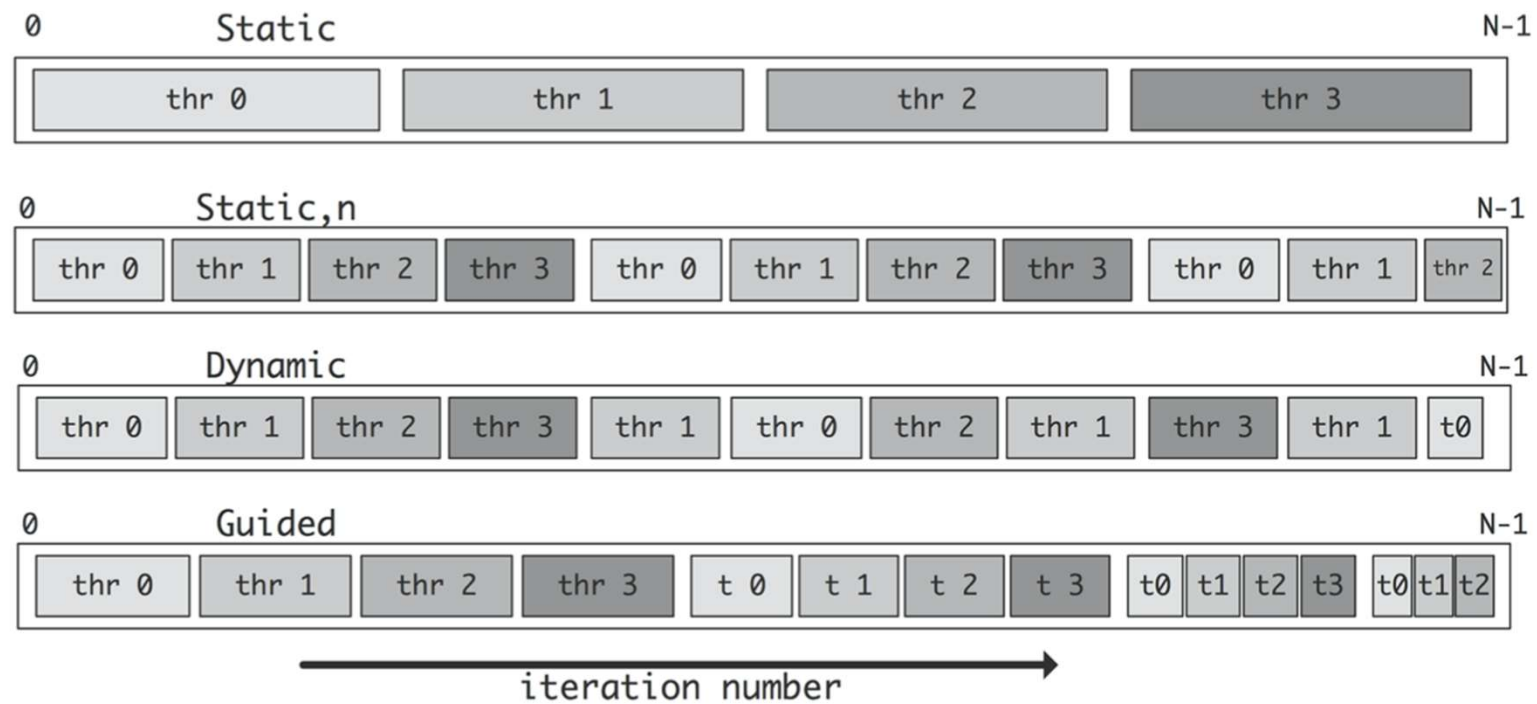


Figure 16.3: Illustration of the scheduling strategies of loop iterations

static

- takes a chunk size (default 1)
- Loop iterations are divided into equal sized pieces of size chunk and then statically assigned to threads. If chunk is not specified, the iteration are evenly (if possible) divided contiguously among the threads



dynamic

- chunk size defaults to 1
- Loop iterations are divided into pieces of size chunk and then dynamically assigned to threads. When a thread finishes one chunk, it is dynamically assigned another.
- must be careful with this, lot of overhead implied here



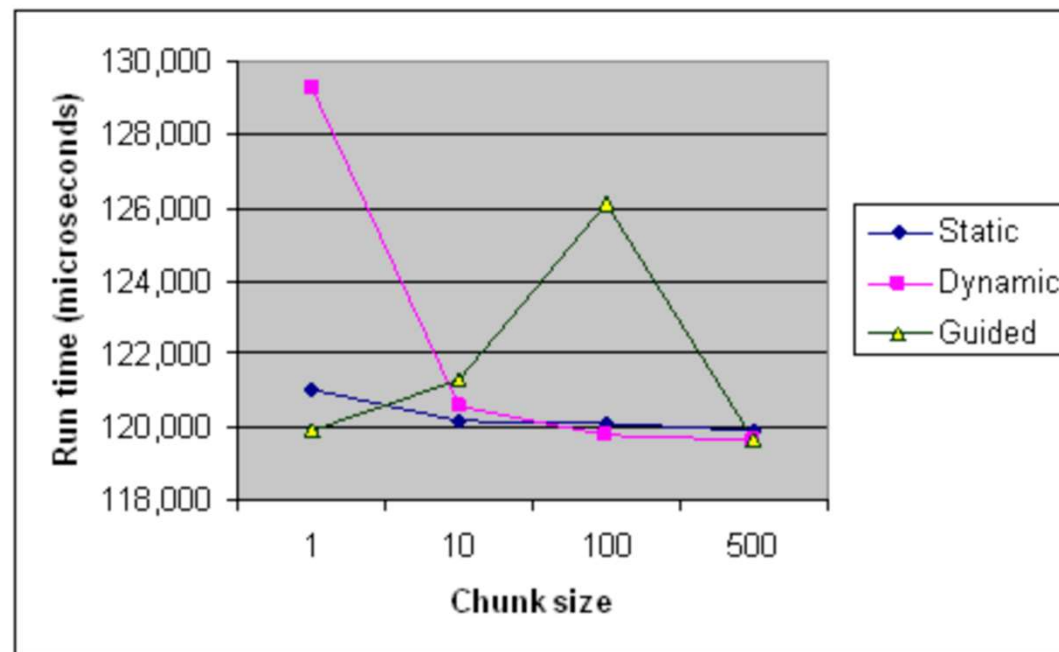
guided

- For a chunk size of 1, the size of each chunk is proportional to the number of **unassigned iterations (dynamic)** divided by the number of threads, decreasing to 1. For a chunk size with value
- $k (k > 1)$, the size of each chunk is determined in the same way with the restriction that the chunks do not contain fewer than k iterations
- even more overhead



It's complicated

- See, e.g.,
<https://stackoverflow.com/questions/42970700/openmp-dynamic-vs-guided-scheduling>



For the progressive problem
What do you get?



By default, worksharing **for** loops end with an implicit barrier

- ***nowait***: If specified, threads do not synchronize at the end of the parallel loop
- ***ordered***: specifies that the iteration of the loop must be executed as they would be in serial program.
- ***collapse***: specifies how many loops in a nested loop should be collapsed into one large iteration space and divided according to the schedule clause. The sequential execution of the iteration in all associated loops determines the order of the iterations in the collapsed iteration space.

COMPUTATIONAL MATH, SCIENCE AND ENGINEERING DEPARTMENT

MICHIGAN STATE
UNIVERSITY

sections and tasks

5312	0000488B	15470B00	004889D6	4889C7E8	6E050000	488B1535	0B000048	89D64889	C7E85C05	0000488D	35790600	00488B05	140B0000	4889C7E8	7C050000	BE080000	..Hä.G...Hä»Hä»En...Hä.5...Hä»Hä»E\...Hç5y...Hä....Hä»E!...ä...
5376	004889C7	E85D0500	00488B15	000B0000	4889D648	89C7E827	050000E8	10050000	4889C348	8D354A06	0000488B	05070A00	004889C7	E83F0500	004889D6	4889C7E8	..Hä»E]...Hä....Hä»Hä»E'...E...Hä/Hç5J...Hä.0...Hä»E?...HäHäHäE
5440	1C050000	488B15C5	0A000048	89D64889	C7E8EC04	0000E8CF	04000048	89C3488D	351E0600	00488B05	9C0A0000	4889C7E8	0A050000	4889D648	89C7E8E1	04000048	...Hä.W...Hä»Hä»E!...Eä...Hä/Hç5...Hä.ü...Hä»E...HäHäHäE...H
5504	8B158A0A	00004889	D64889C7	E8B10400	00488B15	780A0000	4889D648	89C7E89F	04000048	8D35E705	0000488B	05570A00	004889C7	E8BF0400	00BE0800	00004889	ä.ä...Hä»Hä»E±...Hä.x...Hä»Hä»Eü...Hç5A...Hä.W...Hä»Eä...ä...
5568	C7E8A004	0000488B	15430A00	004889D6	4889C7E8	6A040000	488B1531	0A000048	89D64889	C7E85804	0000488D	35B70500	00488B05	100A0000	4889C7E8	78040000	»E†...Hä.C...Hä»Hä»Ej...Hä.1...Hä»Hä»EX...Hç5J...Hä....Hä»EX...
5632	BE040000	004889C7	E8590400	00488B15	FC090000	4889D648	89C7E823	040000E8	F4030000	660F7EC3	488D3588	05000048	8B05D209	00004889	C7E83A04	0000660F	ä...Hä»EY...Hä...Hä»Hä»E#...Eü...f...Hç5a...Hä.«...Hä»E:...f..n/H
5696	6EC34839	C7E8A004	0000488B	15BF0900	004889D6	4889C7E8	E6030000	E8B10300	00660F7E	C3488D35	5B050000	488B0595	09000048	89C7E8FD	03000066	0F6EC348	n/Hä»E...Hä.ö...Hä»Hä»EÉ...E±...f...Hç5J...Hä.i...Hä»E'...f..n/H
5760	89C7E8CD	03000048	8B158209	00004889	D64889C7	E8A90300	00488D35	36050000	488B0561	09000048	89C7E8C9	030000BE	18000000	4889C7E8	9E030000	488B154D	ä»Eö...Hä.Ç...Hä»Hä»Eö...Hç56...Hä.a...Hä»E...ä...Hä»Eü...Hä.M
5824	09000048	89D64889	C7E87403	0000488B	153B0900	004889D6	4889C7E8	62030000	488D3509	05000048	8B051A09	00004889	C7E88203	0000BE08	00000048	89C7E863	...Hä»Hä»E±...Hä.;...Hä»Hä»Eb...Hç5...Hä....Hä»EÇ...ä...Hä»Eö
5888	03800048	8B150609	00004889	D64889C7	E82D0300	00E8E602	00006648	0F7EC348	8D35DA04	C000488B	05D80800	004889C7	E8430300	0066480F	6EC34839	C7E80603	..Hä....Hä»Hä»E-...EÉ...FH...Hç5/...Hä.e...Hä»EÇ...FH.n/Hä»E...
5952	0000488B	15C70800	004889D6	4889C7E8	EE020000	E8A10200	0066480F	7EC3488D	35AC0400	00488B05	9C080000	4889C7E8	04030000	66480F6E	C34889C7	E8C70200	..Hä.«...Hä»Hä»Eö...E'...FH...Hç5...Hä.ü...Hä»E...FH.n/Hä»E...
6016	00488B15	83080000	4889D648	89C7E8AF	02000048	8D358704	0000488B	05670800	004889C7	E8CF0200	00BE3500	00004889	C7E8A402	0000488B	15530800	004889D6	..Hä.ä...Hä»Hä»Eö...Hç5ä...Hä.g...Hä»Eä...ä5...Hä»EÇ...Hä.S...Hä»
6080	4889C7E8	7A020000	488B1541	08000048	89D64889	C7E86802	0000488D	355B0400	00488B05	20080000	4889C7E8	88020000	BE100000	004889C7	E8690200	00488B15	Hä»Ez...Hä.A...Hä»Hä»Eh...Hç5J...Hä...Hä»Eä...ä...Hä»Ei...Hä.
6144	0C080000	4889D648	89C7E83F	89C7E83F	F80B0900	F80B0900	8D353E8B	0000488B	00E07000	89C7E83F	E8690200	004889C7	DB3C2448	89C7E813	02000048	8B15CE07	...Hä»Hä»E3...E'...eJ...Hç53...Hä....Hä»EK...»e<Hä»E...Hä.E.
6208	00004889	D64889C7	E85D0500	00488B15	000B0000	4889D648	89C7E827	050000E8	10050000	4889C348	8D354A06	0000488B	05070A00	004889C7	E83F0500	004889D6	..Hä»Hä»E!...Eÿ...eJ...Hç5...Hä.ö...Hä»E...»e<Hä»E'...Hä.e...
6272	4889D648	89C7E83F	89C7E83F	F80B0900	F80B0900	8D353E8B	0000488B	00E07000	89C7E83F	E8690200	004889C7	DB3C2448	89C7E813	02000048	8B15CE07	004889D6	Hä»Hä»EJ...Hç50...Hä.o...Hä»Eö...ä...Hä»E...Hä.[...Hä»Hä»EÇ...
6336	8B060000	004883C4	485B5DC3	554889E5	4883EC10	897DFC89	75F88370	FC017532	817DF8FF	FF000075	29488D3D	F8070000	E87D0100	00488D15	04E7FFFF	488D35E5	¶...HEFH¶/UHäHäE!..ä..äü'E]...u2A]¶...uHç=...E]...Hç...Hç5Ä
6400	07000048	8B05F606	00004889	C7E86C01	0000C9C3	554889E5	BEFFFF00	00BF0100	0000E8A5	FFFFF5D0	C3554889	ESB80080	FFFF5DC3	554889E5	B8FF7F00	005DC355	...Hä...Hä»E!...UHäÄä...ö...ö...ö.../UHäÄ¶...Ä.../UHäÄ¶...J/U
6464	4889E5B8	00000080	5DC35548	89E5B8FF	FFFF7F5D	C3554889	E5488B00	00000000	0000805D	C3554889	E5488BFF	FFFF7F5D	C3554889	E58B0589	01000066	00000000	HäÄ¶...Ä.../UHäÄ¶...J/UHäÄ¶...Ä.../UHäÄ¶...J/UHäÄä...ä...f
6528	0F6EC05D	C3554889	E58B057D	01000066	0F6EC05D	C3554889	E5488B00	00000000	00100066	480F6EC0	5DC35548	89E548B8	FFFFF7F7	FFFFEF7F	66480F6E	C05DC355	..nJ/UHäÄä...J...f..nJ/UHäÄ¶...FH.nJ/UHäÄ¶...ö...FH.nJ/U
6592	4889E548	B8000000	00000000	00B0A0100	00004889	45F08955	F8DB6D70	5DC35548	89E548C7	C0FFFFF7	FFBAFE7F	00004889	45F08955	F8DB6D70	5DC3FF25	28060000	HäÄ¶¶...Äf...Hä»äü¶»mJ/UHäHäHçJ...J...Hä»äü¶»mJ/V'Ç<...
6656	FF252A06	0000FF25	2C060000	FF252E06	0000FF25	30060000	FF253206	0000FF25	34060000	FF253606	0000FF25	38060000	FF253A06	0000FF25	3C060000	FF253E06	%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...
6720	0000FF25	40060000	FF254206	0000FF25	44060000	FF254606	0000FF25	48060000	FF254A06	0000FF25	4C060000	FF254E06	0000FF25	50060000	FF255206	0000FF25	...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...
6784	54060000	4C8D1D95	05000041	53FF2585	05000090	68000000	00E9E6FF	FFFF6819	000000E9	DCFFFFF7	682B0000	00E9D2FF	FFFF683D	000000E9	00000000	00000000	T...Lç...i...AS'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...'%'...
6848	00E9BEFF	FFFF6861	000000E9	B4FFFFF7	68730000	00E9AAFF	FFFF6885	000000E9	A0FFFFF7	68970000	00E996FF	FFFF68B7	000000E9	8CFFFFF7	68F70000	00E982FF	..Eä...ha...Eÿ...hs...E'...hó...Ej...hó...Eh...hJ...Eä...h...EÇ
6912	FFFF0000	00000000	FFFF777F	FFFF77FF	53697A65	206F6620	73686F72	743A0053	6D616C6C	65737420	73686F72	743A004C	61726765	73742073	686F7274	3A005369	''
6976	7A65206F	6620696E	743A0053	6D616C6C	65737420	696E743A	604C6172	706967374	206967374	3A005369	66206C6F	66E73A00	66206C6F	66E73742	206C6F6E	66C6F6E6	Size of int:Smallest int: Largest int:Size of long:Smallest long
7040	673A004C	61726765	7374206C	6F6E673A	0053697A	65206F66	206C6F6E	67206C6F	6E672069	6E743A00	53697A65	206F6620	666C6F61	743A0053	6D616C6C	65737420	g: Largest long:Size of long long int:Size of float:Smallest float:
7104	666C6F61	743A004C	61726765	7374206C	6C6F6174	3A004469	67697473	20696E20	6D617469	737612C2	20666C6F	61743A00	53697A65	206F6620	646F7562	6C653A00	Float: Largest float: Digits in matissa, float:Size of double:
7168	536D616C	6C657374	20646F75	626C653A	004C6172	7657374	20646F75	626C653A	00446967	69747320	696E206D	61746973	73612C20	646F7562	6C653A00	53697A65	Smallest double: Largest double: Digits in matissa, double:Size
7232	206F6620	6C6F6E67	20646F75	626C653A	00536D61	6C6C6573	74206C6F	6E67206A	6F75626C	653A004C	61726765	7374206C	6F6E6720	646F7562	6C653A00	00000000	of long double:Smallest long double: Largest long double:Size
7296	44696769	74732069	6E206D61	74697373	612C206C	6F6E6720	646F7562	6C653A00	14000000	00000000	017A5200	01781001	100C0708	90010000	34000000	1C000000	Digits in matissa, long double:.....zR.x.....é...4.....
7360	69FCFFFF	FFFF7FFF	0B000000	00000000	00040100	00000E10	86020403	00000000	06040600	00000C07	08000000	00000000	00000000	34000000	54000000	3CFCFFFF	i.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7424	0B000000	00000000	00040100	86020403	00000000	06040600	00000C07	08000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7488	00040100	00000E10	86020403	00000000	06040600	00000C07	08000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7552	86020403	00000000	06040600	00000C07	08000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7616	06040B00	00000C07	08000000	00000000	00000000	34010000	8DFBFFFF	FFFF7FFF	10000000	00000000	00000000	00040100	00000E10	86020403	00000000	06040B00ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7680	08000000	00000000	34000000	6C010000	65FBFFFF	FFFF7FFF	10000000	00000000	00040100	00000E10	86020403	00000000	06040B00	00000C07	08000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7744	34000000	A4010000	3DFBFFFF	FFFF7FFF	10000000	00000000	00040100	00000E10	86020403	00000000	06040B00	00000C07	08000000	00000000	00000000	00000000	4...S...e...ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7808	15FBFFFF	FFFF7FFF	15000000	00000000	00040100	00000E10	86020403	00000000	06041000	00000C07	08000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7872	15000000	00000000	00040100	86020403	00000000	06041000	00000C07	08000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
7936	00040100	00000E10	86020403	00000000	06041A00	00000C07	08000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
8000	86020403	00000000	06041700	00000C07	08000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
8064	06040500	00000303	04600500	000C0708	34000000	F4020000	34F9FFFF	FFFF7FFF	48000000	00000000	00040100	00000E10	86020403	00000000	06044300	00000C07E...ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
8128	08000000	00000000	34000000	2C030000	44F9FFFF	FFFF7FFF	15000000	00000000	00040100	00000E10	86020403	00000000	06041000	00000C07	08000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
8192	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
8256	BF190000	01000000	75190000	01000000	85190000	01000000	4A190000	01000000	3F190000	01000000	65190000	01000000	55190000	01000000	34190000	01000000	ö.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
8320	29190000	01000000	9E1A0000	01000000	9E1A0000	01000000	8A1A0000	01000000	B21A0000	01000000	BC1A0000	01000000	C61A0000	01000000	D01A0000	01000000).....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....ü.....
8384	DA1A0000	01000000	E41A0000	01000000	EE1A0000	01000000	F81A0000	01000000	14190000								

Other worksharing forms

Clearly the for worksharing is a great convenience for an OpenMP program. However there are others that are a little more free form:

- sections
- tasks



Sections

If you know how many elements of work you have to do and would like to divide it up, sections is for you

- use the sections pragma to start off the division
- each section pragma is an independent piece of work that can be run by a thread in the team



Tasks

- first appeared in OpenMP 3
- a kind of dynamic, independent scheduling of tasks that get assigned to threads.
- allows for recursion.



dynamic

With sections, you have to know how many you have.

With tasks, you are "generating" a dynamic list of tasks that get thread scheduled.



```

int main (){
#pragma omp parallel num_threads(2) default(none) shared(cout)
{
#pragma omp task
{
    stringstream s;
    s << "Doing task 1 on thread" << omp_get_thread_num() << endl;
    cout << s.str();
}

#pragma omp task
{
    stringstream s;
    s << "Doing task 2 on thread" << omp_get_thread_num() << endl;
    cout << s.str();
}

} // of parallel
}

```

```

>./a.out
Doing task 1 on thread0
Doing task 1 on thread0
Doing task 2 on thread0
Doing task 2 on thread1
(base) [13:36][545][bill@
>./a.out
Doing task 1 on thread0
Doing task 2 on thread1
Doing task 2 on thread1
Doing task 1 on thread0
(base) [13:36][546][bill@
>./a.out
Doing task 1 on thread0
Doing task 1 on thread0
Doing task 2 on thread0
Doing task 2 on thread1

```


two weirdness-es

1. 4 outputs, only two tasks: why?
2. output order, task assignment to thread changes each time



2)

Scheduling of thread to task is up to the implementation

Output order is always screwed up



4 outputs?

The way this is coded, each thread encounters the parallel section so each thread schedules two tasks.

Probably not what we wanted.



```

int main (){
#pragma omp parallel num_threads(2) default(none) shared(cout)
{
#pragma omp single
{
#pragma omp task
{
stringstream s;
s << "Doing task 1 on thread" << omp_get_thread_num() << endl;
cout << s.str();
}

#pragma omp task
{
stringstream s;
s << "Doing task 2 on thread" << omp_get_thread_num() << endl;
cout << s.str();
} // of single
} // of parallel
}

```

```

>./a.out
Doing task 2 on thread0
Doing task 1 on thread1
(base) [13:45][550][bill@msu ~]$
>./a.out
Doing task 2 on thread0
Doing task 1 on thread1
(base) [13:45][551][bill@msu ~]$
>./a.out
Doing task 2 on thread0
Doing task 1 on thread1
(base) [13:45][552][bill@msu ~]$
>

```