HW 11 Solution

Find the isoefficiency of the following program under the assumptions given below:

```
r = 0;
for (i=0; i < n; i++) {
   r = r + a[i];
}
```

1. When the recurrence is written as:

```
double r[threadCount];
s = 0;
#pragma omp parallel for
for (i=0; i < n; i++) {
   r[omp_get_thread_num( )] = r[omp_get_thread_num( )] + a[i];
}
for (i=0; i < threadCount; i++) {
   s += [i];
}
```

Solution: serial time is n-1, which we will simplify to n. Let T=threadCount. The parallel time is

$n/T + T$. The parallel overhead is $T^*(n/T+T) - n = n + T^2 - n = T^2$. Thus $W(n) = T^2$, which is the isoefficiency function, and therefore work must increase as the square of the number of threads to maintain the same efficiency with a larger number of threads. Thus going from 4 to 16 threads goes from 16 to 256 units of work needed, or an increase of 16X work.

2. When the recurrence is written as:

```
#pragma omp parallel for reduction(+:r)
for (i=0; i < n; i++) {
   r = r + a[i];
}
```

Solution: The parallel time is $n/T + \log_2 T$. The parallel overhead is $\mathbf{T^*(n/T + \log_2 T) - n}$ = $n + T \log_2 T - n = T \log_2 T$. Thus $W(n) = T \log_2 T$, , which is the isoefficiency function, and therefore work must increase as $T \log_2 T$ maintain the same efficiency with a larger number of threads. Thus going from 4 to 16 threads goes from 8 to 64 units of work needed, or 8X more work. **Note:** I am not counting communication cost here, only adds. If we count communication costs, the expressions $T^*(n/T + \log_2 T) - n$ in bold above should become $T^*(n/T + \mathbf{2} \log_2 T) - n$, where the one $\log_2 T$ term is the adds and one $\log_2 T$ is the communication through shared memory.

3. When the recurrence is written as:

```
#pragma omp parallel for simd reduction(+:r)
for (i=0; i < n; i++) {
   r = r + a[i];
}
```

In this program, assume that it is like the program in 2 except that four adds are done for the cost of one add when finding the partial sums on each thread, and when combining the partial sums across threads four partial sums can be added each time.

Solution: The parallel time is n/)T*4) + $\log_4 T$. The parallel overhead is

$T*(n/(T*4) + \log_4 T) - n$
$= -n/4 + T \log_4 T - n =$
$-3/4n + T \log_4 T$.

Thus $W(n) = -3/4n + T \log_4 T$. Solving for n we get
$n = -3/4n + T \log_4 T$
$7n/4 = T \log_4 T$
n = **(4/7) T $\log_4$T**, which is the isoefficiency function, and therefore work must increase as (4/7) T $\log_4$T to maintain the same efficiency with a larger number of threads. Thus going from 4 to 16 threads goes from (4/7) * 4 * 1 = 2.3 or 3 units of work to (4/7) 16 * 2 = 18.3 or 19 units of work needed, or ~6X more work. **Note** that since in the four thread case the concurrency is limited we would really need 4 units of work to have at least one unit of work per thread, and so we would do ~5X times more work at 16 threads. At T > 4 this is not an issue. **Note:** I am not counting communication cost here, only adds. If we count communication costs, the expressions (4/7) T $\log_4$T in bold above would become (4/7) 2 T $\log_4$T (or (8/7) T $\log_4$T), where the one $\log_4$T term is the adds and one $\log_4$T is the communication through shared memory.

4. Compare the work involved in the recurrence of 1, 2 and 3 at 4 threads, 16 threads, 64 threads and at 256 threads using the iso-efficiency relationships you came up with as answers.

T / isoefficiency function	T^2	$T \log_2 T$	$(4/7)\ T \log_4 T$
4	16	8	3 (or 4, as explained above)
16	256	64	19
64	4096	64 * 5 = 320	(4*64/7) * 3 = ~110
256	65536	2048	(4*256/7) * 4 = ~585