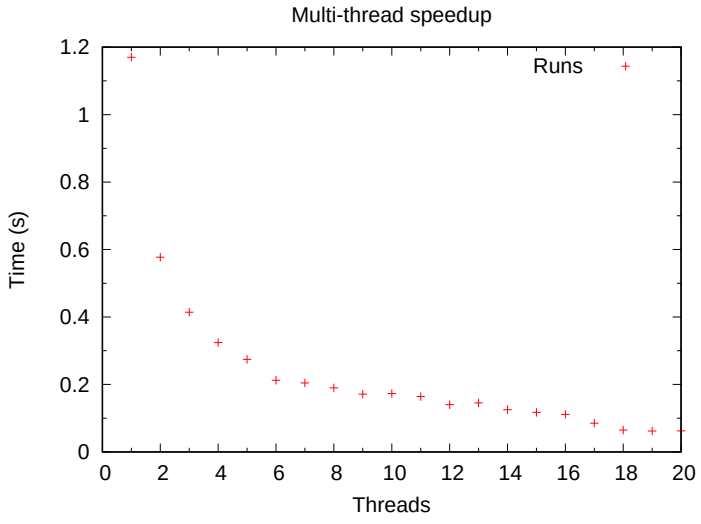
N	Result
10000000.000000000	50000022847696.773

N	Result	Time(sec)	Threads
10000.000000000000	50022902.063502021	2.7233389992034063E-003	1
20000.000000000000	200045747.49332842	1.2078340005245991E-003	1
50000.000000000000	1250114373.2691557	4.2834539999603294E-003	1
100000.00000000000	5000228702.5078430	7.3718460007512476E-003	1
200000.00000000000	20000457210.329472	1.5385407999929157E-002	1
500000.00000000000	125001142626.86575	3.5706549000678933E-002	1
1000000.0000000000	500002285262.38922	7.0625592999931541E-002	1
2000000.0000000000	2000004569634.1506	0.13367078699957347	1
5000000.0000000000	12500011423818.637	0.32866591200036055	1
10000000.000000000	50000022847696.773	0.70946930500031158	1
20000000.000000000	200000045696542.53	1.4798242030010442	1
50000000.000000000	1250000114230899.5	4.8390261109998391	1

N	Result	Time(sec)	Threads
10000000.000000000	50000022847696.773	1.1699680499732494	1
10000000.000000000	50000022847758.734	0.57710395986214280	2
10000000.000000000	50000022847758.633	0.41409664507955313	3
10000000.000000000	50000022847751.398	0.32401113817468286	4
10000000.000000000	50000022847746.953	0.27427113894373178	5
10000000.000000000	50000022847745.266	0.21259477827697992	6
10000000.000000000	50000022847744.742	0.20447296509519219	7
10000000.000000000	50000022847743.711	0.18985434295609593	8
10000000.000000000	50000022847742.219	0.17137576499953866	9
10000000.000000000	50000022847742.484	0.17297412827610970	10
10000000.000000000	50000022847742.086	0.16433870512992144	11
10000000.000000000	50000022847742.039	0.14020182890817523	12
10000000.000000000	50000022847742.273	0.14532390516251326	13
10000000.000000000	50000022847742.945	0.12497906899079680	14
10000000.000000000	50000022847742.812	0.11712360242381692	15
10000000.000000000	50000022847742.898	0.11148337274789810	16
10000000.000000000	50000022847742.336	8.5268609691411257E-002	17
10000000.000000000	50000022847742.117	6.4955320674926043E-002	18
10000000.000000000	50000022847742.438	6.1813935171812773E-002	19
10000000.000000000	50000022847742.977	6.2402177136391401E-002	20



Significant speedup when going from 1 to 4 threads as balance of workload sharing through the parallelisation changes. At higher thread counts the bookkeeping required when using parallel approaches begins to balance the parrallel speedup, so there is less of a performace increase with each subsiquent thread.

OMP file

```
program q6_ac2071
use omp_lib
use omp_lib_kinds
implicit none

integer,parameter                    :: dp= selected_real_kind(15,300)
integer,parameter                    :: li= selected_int_kind(16)
real(kind=dp), dimension(12)         :: N_array=
(/10e+3,2*10e+3,5*10e+3,10e+4,2*10e+4,5*10e+4,10e+5,2*10e+5,5*10E+5,10e+6,&
2*10e+6,5*10e+6/)
! real(kind=dp), dimension(1)         :: N_array= (/10e+6/)
real(kind=dp), dimension(:),allocatable  :: A,B
real(kind=dp)                        :: N
integer(kind=li)                     :: i, j,threads, int_N
real(kind=dp)                        :: start_time=0.0_dp, end_time=0.0_dp,total_sum

!loop over all N in N_array
do j=1,size(N_array)
   !select N
   N=N_array(j)
   int_N=int(N,kind=li)

   allocate(A(int_N))
   allocate(B(int_N))
   total_sum=0.0_dp
   !$OMP parallel private(i) shared (int_N,A,B) reduction(+:total_sum)
   start_time = omp_get_wtime()
   threads = omp_get_num_threads()

   !carry out sum
   !$OMP do
   do i=1,int_N
      !removed dependencies
      A(i) = real(i,kind=dp)
      B(i) = A(i)*A(i)
      A(i) = A(i) + sqrt(abs(sin(B(i))))*2.34_dp
      !$OMP atomic
      total_sum = total_sum + A(i)
   enddo
   !$OMP end do
   end_time = omp_get_wtime()
   !$omp end parallel
```

```
    print*,N, total_sum, (end_time-start_time),threads
    deallocate(A)
    deallocate(B)

enddo

endprogram
```

Shell script

```
#!/bin/sh

threads="1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20"
# threads="1 2 3 4"

#loop over threads
for T in $threads
do
    export export OMP_NUM_THREADS=$T
    #execute code
    ./q6_ac2071.exe
done
```