# Environment Setup

OPENMP DISPLAY ENVIRONMENT BEGIN

\_OPENMP = '201511'

OMP\_DYNAMIC = 'FALSE'

OMP\_NESTED = 'TRUE'

OMP\_NUM\_THREADS = '10'

OMP\_SCHEDULE = 'STATIC'

OMP\_PROC\_BIND = 'SPREAD'

OMP\_PLACES = '{0:10}'

OMP\_STACKSIZE = ‘16777216'

OMP\_WAIT\_POLICY = 'PASSIVE'

OMP\_THREAD\_LIMIT = '4294967295'

OMP\_MAX\_ACTIVE\_LEVELS = '255'

OMP\_CANCELLATION = 'TRUE'

OMP\_DEFAULT\_DEVICE = '0'

OMP\_MAX\_TASK\_PRIORITY = '0'

OMP\_DISPLAY\_AFFINITY = 'FALSE'

OMP\_AFFINITY\_FORMAT = 'level %L thread %i affinity %A'

OMP\_ALLOCATOR = 'omp\_default\_mem\_alloc'

OMP\_TARGET\_OFFLOAD = 'DEFAULT'

OPENMP DISPLAY ENVIRONMENT END

# Performance measurement & Explanation

Sorting 100000000 elements of type int (381.000000 MiB)...

done, took 9.713000 sec of serial

Verification... successful.

done, took 1.935000 sec of parrallel.

Verification... successful.

Because of the iteration of recursion is unknown, so we used the omp task method for parallelization.

Use depth limited mechanism to prevent too much task generated to block threads.

When the array size <= 800, it is more efficient sort by serial, so we used the cutoff mechanism to get better. performance.

Use omp\_get\_max\_threads() function to get threads dynamically so that it can perform well in different computer.

In the merge sort algorithm, the scale of each recursion is similar, so we set schedule to static to avoid unnecessary operate.

Setting the stack size to 16M prevents the stack from running out of space.