Parallel Loops: Merge Sort

Question: Before starting, run all sequential codes in Centaurus using `make bench`.

1 Merge Sort

Question: Implement a parallel function using parallel loop constructs to perform merge sort on an array of integer. Use the template of `mergesort/mergesort.cpp`. Note that the data is generated by function `generateMergeSortData()` and the results is checked by `checkMergeSortResult()`. Remember to set thread count and granularity using the `setNbThread()` and `setGranularity()` functions provided in the `omploop.hpp` file. Output the time it took on `stderr`.

Note: MergeSort is clearly a recursive algorithm. But to use parallel looping construct, you will have to rewrite Merge Sort first as a iterative algorithm. A good way to think about it is to start from the dependencies of merge sort and identify how to express that dependency structure using parallel for-loops while discounting its natural recursive writing.

Question: Run the code on Centaurus, in the `mergesort/` directory, using `make bench`. And then plot the results using `make plot`. Does the plot make sense? Why?

Question: (Extra Credit) Still using only Parallel loops, make Merge Sort more parallel by making Merge parallel.