MapReduce is based on (on occasion, randomly) segregating the data into different sub-problems (Mapping), computationally process each of these sub-problems in parallel, and then merging the computation results in a single step so as to answer the problem that was initially posed for the data (Reducing). It can be considered a special case of divide and conquer, which involves a reduction (via filtering/merging) in the final set of computations generated at the end of the mapping process. The divide and conquer algorithm essentially means that we divide a problem into sub-problems, and keep applying this technique at each split until the individual sub-problems can be solved independently; we then combine these sub-problems in a bubble-up approach (from the bottom of the tree to the top, towards the original problem) to answer the question that was asked. The modification to the divide and conquer algorithm, in MapReduce, is that all the subproblems need to be computationally resolved into compact solutions (ex. Sorting the sub-solutions, or hashing them, to make associations with other results quicker and easier) that can be put together by a single processor at the end (i.e. the Reduce step). Furthermore, in MapReduce, the initial separation of data into sub-problems, while recursive in Divide and Conquer, can be initially dictated by other things (allocation by the scheduler, for ex.) and then be recursive.