Fast Parallel Sorting Algorithms

Summary

A parallel bucket-sort algorithm is that requires time O(log n) and the use of n processors. A procedure is also presented to sort n numbers in time O(k log n) using n 1+1/k processors, for k an arbitrary integer. In parallel algorithms, there is a similar tradeoff between time and the number of processors used. problem using a bounded number of processors, a minimal amount of time is required. We shall present an algorithm for sorting n numbers in time O(log n) that requires asymptotically fewer processors. We first present parallel bucket-sorting algorithms in which, at the expense of greater space requirements, the number of processors and the amount of time used are both reduced. The algorithms are unusual in that the space requirements are greater than the processor time requirements. Representation of the processor number, the processors being numbered by consecutive integers starting from zero. This is a parallel version of the "bucket sort. “An obvious implementation of the parallel bucket sort would be for each processor pi (which has temporarily been "assigned to" cz, the ith number being sorted) to place the value i in bucket ci. Then, for each number appearing among the numbers being sorted, only one processor (the one with smallest index) will be active when we place i in bucket c~. If the buddy is not active or if it is active but is of higher rank, then the processor will continue, shifting its mark to the location of the buddy if that location is of lower index than the one  
currently in use. After the kth iteration, a mark will be present at each location whose last k bits are zeros and who’s other (log n) - k bits coincide with the corresponding bits of the address of a processor active in the same area. Thus, each such location will be marked iff any of 2 k processors had been active in that area originally. After log n iterations, the first location in an area will be marked iff any of the n processors originally were active in that area, i.e., iff any of the n numbers to be sorted was j, the  
area bucket number. It is noted that this bucket-sort algorithm requires space S = O(mn), time T -- O(log n), and the use of n processors. There is another bucket sort algorithm in which However, instead of a simple mark bit, we shall keep a running count of how many processors were originally active in each block of indices  
of size 2 k. In any case, all processors pi (active or not) will add to their running  
count of the number of processors (that were originally active) having indices greater than i. Active processors keep their count at the head of the largest block that they have investigated (which will be of size 2k).