Abstract

A general method for generating 3-dimensional sorting algorithms by using 2-dimensional algorithms is presented. The main advantage is that from a large class of sorting algorithms suitable for mesh-connected rectangles of processors we efficiently obtain sorting algorithms suitable for 3-dimensional meshes. It is shown that by using the $s^2$-way merge sort of Thompson and Kung sorting $n^3$ elements can be performed on an $n \times n \times n$ cube with $12n + 0( n^{2/3} \log n )$ data interchange steps. Further improvements lead to an algorithm for an $n/2 \times n \times 2n$ mesh sorting $n^3$ items within $10.5n + O( n^{2/3} \log n )$ interchange steps. By a generalization of the method to $r$-dimensional cubes one can obtain algorithms sorting $n^r$ elements with $O(r^3n)$ interchange steps.

1. Introduction

The design and analysis of fast parallel algorithms has become more and more important by the advancements of VLSI-technology. Especially for VLSI-architectures, where a regular net of simple processing cells and local communication between these cells are required [FK,KL], several parallel algorithms for fundamental problems as matrix arithmetic, signal and image processing, sorting and searching etc. have been proposed [U].

In this paper a general method for generating 3-dimensional sorting algorithms by using 2-dimensional algorithms is presented. The advantage compared with former research [TK,NS] is that from a large class of sorting algorithms suitable for mesh-connected rectangles of processors [KH,NS,LSSS,SI,TK] we efficiently obtain sorting algorithms suitable for 3-dimensional meshes.

*This work was partially done at the Institut für Informatik, University of Kiel, and partially supported by the Siemens AG, München.
A mesh-connected $n_1 \times n_2 \times \ldots \times n_r$ array of processors is a set of $N = n_1 n_2 \ldots n_r$ identical processors where each processor $P = (p_1, \ldots, p_r)$, $1 \leq p_i \leq n_i$, is directly interconnected to all its nearest neighbours only. A processor $Q = (q_1, \ldots, q_r)$ is called a nearest neighbour of $P$ if and only if the distance fulfills $d(P,Q) = \sum_{i=1}^r |p_i - q_i| = 1$. For example, for $r = 2$, that is in the plane, every processor has at most 4 nearest neighbours. Note that no ”wrap-around” connections are allowed. At each time step each processor can only communicate with one of its nearest neighbours. That is, at most $N/2$ communications can simultaneously be performed. For the sorting problem we assume that $N$ elements from a linearly ordered set are loaded in the $N$ processors, each receiving exactly one element. The processors are thought to be indexed by a certain one-to-one mapping from $\{1, \ldots, n_1\} \times \ldots \times \{1, \ldots, n_r\}$ onto $\{1, \ldots, N\}$. With respect to this function the sorting problem is to move the $i$-th smallest element to the processor indexed by $i$ for all $i = 1, \ldots, N$.

In the following for the 3-dimensional case we assume an index function $f$ with $f(p_1,p_2,p_3) = n_1 n_2 (p_3 - 1) + g(p_1,p_2)$, where $g$ is either a pure or a snake-like or a shuffled row-major indexing for an $n_1 \times n_2$ mesh [TK] (Figure 1). If $g$ is a (pure) row-major index function, then one might call $f$ a plane-major-row-major indexing [S] (Figure 2).

Clearly, the sorting problem can be solved by a sequence of comparison and interchange steps. It is well-known that data movement is a significant performance measure for sorting algorithms on mesh-connected architectures. Therefore, in this paper we concentrate on the number of data interchange steps which may be caused by a comparison or not. Note that one interchange step is equivalent to two routings in [TK].

For the 3-dimensional case sorting algorithms for an $n \times n \times n$ mesh-connected cube have already been proposed in [TK,NS]. Both algorithms asymptotically need $15n$ interchange steps whereas Schimmler [S] recently developed a simpler sorting algorithm on a cube with $19n$ interchange steps. All the algorithms are generalizations of special 2-dimensional sorting algorithms and use recursion steps where eight presorted $n/2 \times n/2 \times n/2$ cubes are merged to one sorted $n \times n \times n$ array.

In the second section of this paper we present a method (called 3BY2) for obtaining sorting algorithms on arbitrary $a \times b \times c$ arrays based on arbitrary sorting algorithms for mesh-connected rectangles with row-major indexing. For a 2-dimensional mesh-connected $u \times v$ rectangle let $\text{SORT}(u,v)$ denote the number of interchange steps needed by the underlying sorting algorithm $\text{SORT}$. If for an $a \times b \times c$ mesh $3D-\text{SORT}(a,b,c)$ denotes the corresponding number for that 3-dimensional algorithm which have been obtained by an application of the method 3BY2 to algorithm $\text{SORT}$, then it is shown that

$$3D-\text{SORT}(a,b,c) \leq \text{SORT}(a,c) + \text{SORT}(b,c) + 2 \cdot \text{SORT}(a,b) + 2.$$