Cost-Optimal Parallel Algorithms for Constructing B-Trees

BIING-FENG WANG

and

GEN-HUEY CHEN

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, Republic of China

Communicated by Tosiyasu L. Kunii.

ABSTRACT

In this paper two cost-optimal parallel algorithms are presented for constructing a B-tree for a sorted list of N keys. These two parallel algorithms are designed on the shared-memory SIMD computer: one, based on the EREW model, uses $N/\log\log N$ processors and requires $O(\log\log N)$ time; the other, based on the CREW model, uses N processors and requires $O(1)$ time.

1. INTRODUCTION

Search trees [3, 8], such as binary search trees, m-way search trees, 2-3 trees, and B-trees are efficient data structures to store a sorted list of keys (here keys may denote records). Search, insertion, and deletion with respect to a balanced search tree of N keys can be done in worst-case time $O(\log N)$. The problem of constructing a balanced search tree for a sorted list of N keys is to create a balanced search tree with minimal height to store the sorted list. To create such a tree, the keys within each node must be determined and the links between nodes must be set up properly.

Correspondence to Professor Gen-Huey Chen, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, Republic of China.

Optimal sequential algorithms for constructing balanced binary search trees can be found in [4] and [6]. Based on the EREW (exclusive read, exclusive write) shared-memory model, Moitra and Iyengar [9, 10] have designed parallel algorithms that use N processors to construct balanced binary search trees in $O(1)$ time. When there are too many keys to be kept in the main memory, they must be saved as an external file. In such a situation, it is helpful to keep more than one key in each node in order to balance key processing time and file accessing time. Therefore, m-way search trees are suitable data structures for storing a large number of keys. In [7], also based on the EREW shared-memory model, Dekel et al. have designed parallel algorithms that use N processors to construct balanced m-way search trees. The time complexities of Dekel et al.'s algorithms are $O(1)$ under an assumption that computing the kth, $0 \leqslant k \leqslant \log N$, power of m takes $O(1)$ time.

The main disadvantage of the m-way search tree is that it may become unbalanced when it is not static [1, 8], and thus logarithmic access time cannot be guaranteed. For example, let us consider a sequence of update operations such as insertion and deletion on a balanced m-way search tree. The tree may become unbalanced after these operations are completed and then the access time of the tree may become proportional to the tree size in the worst case. Therefore, periodic rebalancing is indispensable to an m-way search tree after some update operations are performed on it. This leads to the unsuitability of on-line processing of the m-way search tree, especially when it is stored in the auxiliary memory. As an alternative to the m-way search tree, some other trees have been proposed, for example, AVL tree [8], weight-balanced tree [8], and B-tree [2, 5]. Among them, the B-tree, which was originally proposed by Bayer and McCreight [2], is the most attractive one for a number of reasons. First, periodic rebalancing is unnecessary. Second, logarithmic access time is guaranteed for on-line processing. Third, insertion and deletion to the B-tree are quite simple. Fourth, as compared with the AVL tree and the weight-balanced tree, the B-tree is free of keeping additional information for maintaining itself balanced. Finally, in addition to being an alternative representation of a sorted list, the B-tree has the capabilities of concatenating and splitting lists. A more recent description of B-trees and some of their variations can be found in [5].

Recently, Wang and Chen [11] designed parallel algorithms for constructing 2-3 trees, which are B-trees of order 3 (explained in the next section). In this paper, as an extension of their results, we present two cost-optimal parallel algorithms for constructing B-trees of arbitrary order m. The two parallel algorithms are designed on the shared-memory SIMD (single instruction stream, multiple data stream) computer: one, based on the EREW model, uses $N/\log\log N$ processors and requires $O(\log\log N)$ time; the other, based on the CREW (concurrent read, exclusive write) model, uses N processors and requires $O(1)$ time, where N is the number of keys stored in the B-tree. Like [7], [9], [10], and [11], we assume that computing the logarithm of a number k takes $O(1)$ time.

The rest of this paper is organized as follows. In the next section, notation and definitions that are used throughout this paper are introduced. In Section 3, the shape of the B-tree to be constructed for a given value of N is uniquely specified. Also, some properties that are relevant to our construction algorithms are described. Then, in Section 4, the two parallel construction algorithms are presented. Finally, concluding remarks are given in Section 5.

2. NOTATION AND DEFINITIONS

DEFINITION 1. A *B-tree of order m*, $m \geqslant 3$, is a tree satisfying the following properties.

1. Every node has at most m children.
2. Every node, except for the root, has at least $\lceil m/2 \rceil$ children.
3. The root has at least two children.
4. All leaves are on the same level.
5. A node with w children contains $w-1$ keys.

For the convenience of description, we use notation $\langle i,j \rangle$ to denote the jth node from the left at the ith level (see Figure 1). For example, the root node is denoted by $\langle 1,1 \rangle$ and its left child is denoted by $\langle 2,1 \rangle$. A node $\langle i,j \rangle$ with w keys and $w+1$ pointers can be represented as

$$(P_{\langle i,j\rangle,1}, K_{\langle i,j\rangle,1}, P_{\langle i,j\rangle,2}, K_{\langle i,j\rangle,2}, \ldots, P_{\langle i,j\rangle,w}, K_{\langle i,j\rangle,w}, P_{\langle i,j\rangle,w+1}),$$

where $K_{\langle i,j\rangle,1} < K_{\langle i,j\rangle,2} < \cdots < K_{\langle i,j\rangle,w}$ denote the w keys, $P_{\langle i,j\rangle,1}$ points to the subtree whose keys are smaller than $K_{\langle i,j\rangle,1}$, $P_{\langle i,j\rangle,v}$, $1 < v < w+1$, points to the subtree whose keys are between $K_{\langle i,j\rangle,v-1}$ and $K_{\langle i,j\rangle,v}$, and $P_{\langle i,j\rangle,w+1}$ points to the subtree whose keys are greater than $K_{\langle i,j\rangle,w}$. If node $\langle i_1,j_1 \rangle$ and node $\langle i_2,j_2 \rangle$ are contained, respectively, in two subtrees pointed by $P_{\langle i,j\rangle,v}$ and $P_{\langle i,j\rangle,v+1}$, $1 \leqslant v \leqslant w$, then the key $K_{\langle i,j\rangle,v}$ is said to be the split key of node $\langle i_1,j_1 \rangle$ and node $\langle i_2,j_2 \rangle$. For example, let us consider Figure 1a, where $P_{\langle 2,2\rangle,1} = \langle 3,4 \rangle$, $P_{\langle 2,2\rangle,2} = \langle 3,5 \rangle$, and $P_{\langle 2,2\rangle,3} = \langle 3,6 \rangle$. The split key of nodes $\langle 4,4 \rangle$ and $\langle 4,5 \rangle$ is 12.

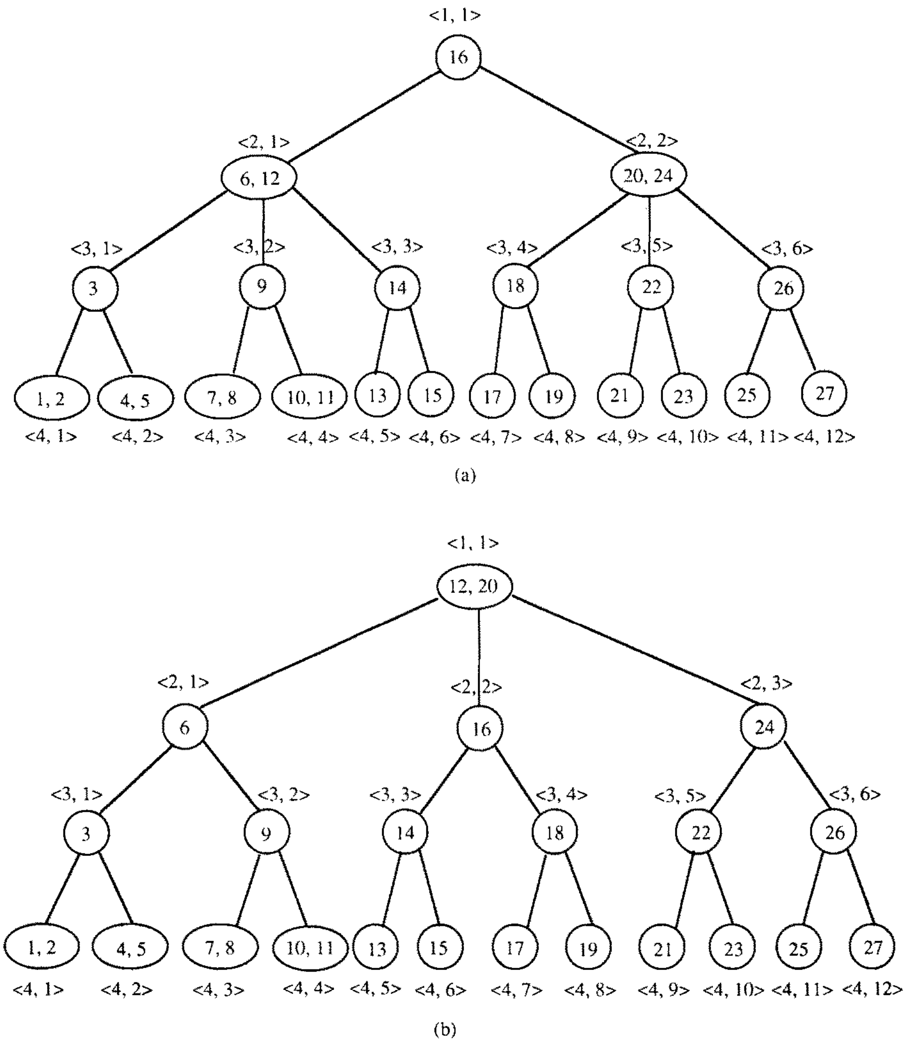

Fig. 1. Two different B-trees of order 3 that store the sorted list $(1,2,3,\ldots,26,27)$.

Note that more than one B-tree of order m can store a given sorted list. For example, Figure 1 shows two different B-trees of order 3 that can store the sorted list $(1,2,3,\ldots,26,27)$. Thus, before presenting the parallel construction algorithms, it is helpful to specify the exact one that our algorithms will construct for a given sorted list. For a given sorted list of size

N, the unique B-tree of order m to be constructed has the following properties:

1. It has the minimal height $n = \lceil \log_m(N+1) \rceil - 1$.
2. The root owns $r = \lceil (N+1)/m^n - 1 \rceil$ keys. Note that $m^n \leqslant N \leqslant m^{n+1} - 1$; hence, $1 \leqslant r \leqslant m - 1$.
3. There exists a unique integer c, $1 < c \leqslant n+1$, such that all the nodes, except the root, above the cth level own $m-1$ keys and all the nonleaf nodes on and below the cth level own $\lceil m/2 \rceil - 1$ keys.
4. Let s be the number of keys contained in the leaf node $\langle n+1, 1 \rangle$, $\lceil m/2 \rceil \leqslant s \leqslant m-1$. Then, each of the other leaf nodes may own s or $s-1$ keys, but may not own more keys than the leaf nodes on its left.

For example, the B-tree of order 3 that our algorithms will construct for the sorted list $(1,2,3,\ldots,26,27)$ is shown in Figure 1a. In this example, $n = 3$, $r = 1$, $c = 3$, and $s = 2$.

The existence and uniqueness of the B-tree with properties 1–4 is shown in the next section. In the following, we define notations that are used throughout this paper:

m	The order of the B-tree to be constructed. Note that $m \geqslant 3$ is a constant.
N	The number of keys in the given sorted list. We assume $N > m$.
$T_{N,m}$	The B-tree of order m to be constructed for a sorted list of N keys.
n	The height of $T_{N,m}$, which is defined as the number of levels of $T_{N,m}$. Note that $n = \lceil \log_m(N+1) \rceil - 1$.
r	The number of keys that are contained in the root of $T_{N,m}$. Note that $r = \lceil (N+1)/m^n - 1 \rceil$.
c	The marked level of $T_{N,m}$. That is, in $T_{N,m}$, all the nodes, except the root, above the cth level own $m-1$ keys and all the nonleaf nodes on and below the cth level own $\lceil m/2 \rceil - 1$ keys. Note that $1 < c \leqslant n+1$.
$V_{N,m}$	The number of nodes in $T_{N,m}$.
$L_{N,m}$	The number of leaf nodes in $T_{N,m}$. Note that, according to the construction of $T_{N,m}$, $L_{N,m} = (r+1) * m^{c-2} * \lceil m/2 \rceil^{n-c+1}$.
$W_{N,m}$	The total number of keys that are contained in the leaf nodes of $T_{N,m}$. Note that $L_{N,m} * (\lceil m/2 \rceil - 1) + 1 \leqslant W_{N,m} \leqslant L_{N,m} * (m-1)$.
s	The number of keys that are contained in the leaf node $\langle n+1, 1 \rangle$ of $T_{N,m}$. Note that $\lceil m/2 \rceil \leqslant s \leqslant m-1$.

$LL_{N,m}$	The number of leaf nodes in $T_{N,m}$ that own s keys.
$RANK_{N,m}(K_{\langle i,j\rangle,v})$	The rank of the key $K_{\langle i,j\rangle,v}$ in the given sorted list. It equals the number of keys (inclusive of $K_{\langle i,j\rangle,v}$ itself) that precede $K_{\langle i,j\rangle,v}$ in in-order traversal of $T_{N,m}$. Note that if node $\langle i,j\rangle$ contains w keys, $RANK_{N,m}(K_{\langle i,j\rangle,v})$ is undefined for $v > w$.
$PREC_{N,m}(K_{\langle i,j\rangle,v})$	The number of leaf nodes in $T_{N,m}$ that precede the key $K_{\langle i,j\rangle,v}$ in in-order traversal of $T_{N,m}$. Note that if node $\langle i,j\rangle$ contains w keys, $PREC_{N,m}(K_{\langle i,j\rangle,v})$ is undefined for $v > w$.

For example, let us consider Figure 1a again, where $m=3$, $N=27$, $n=3$, $r=1$, $c=3$, $V_{27,3}=21$, $L_{27,3}=12$, $W_{27,3}=16$, $s=2$, $LL_{27,3}=4$, $RANK_{27,3}(K_{\langle 2,1\rangle,1})=6$, $RANK_{27,3}(K_{\langle 2,1\rangle,2})=12$, $PREC_{27,3}(K_{\langle 2,1\rangle,1})=2$, and $PREC_{27,3}(K_{\langle 2,1\rangle,2})=4$.

3. SOME PROPERTIES

Two graphs G and G' are said to be isomorphic if there is a one-to-one correspondence between their vertices and between their edges such that the incidence relationship is preserved. In other words, suppose that edge e is incident on vertices v_1 and v_2 in G; then the corresponding edge e' in G' must be incident on the vertices v_1' and v_2' that correspond to v_1 and v_2, respectively. In this section, considering all isomorphic B-trees equivalent, we prove that $T_{N,m}$ uniquely exists. In addition, we introduce some properties of $T_{N,m}$, which are the kernel of our construction algorithms.

LEMMA 1. *The B-tree $T_{N,m}$ is uniquely specified by a pair of c and $W_{N,m}$, where $1 < c \leqslant n+1$ and $L_{N,m} * (\lceil m/2\rceil - 1) + 1 \leqslant W_{N,m} \leqslant L_{N,m} * (m-1)$.*

Proof. Clearly, according to the construction of $T_{N,m}$, $T_{N,m}$ is uniquely specified by a triplet of c, s, and $LL_{N,m}$, where $1 < c \leqslant n+1$, $\lceil m/2\rceil \leqslant s \leqslant m-1$, and $LL_{N,m} \geqslant 1$, because the height of $T_{N,m}$ is $n = \lceil \log_m(N+1)\rceil - 1$ and the number of keys that are contained in the root is $r = \lceil (N+1)/m^n - 1\rceil$. Given a pair of c and $W_{N,m}$, where $1 < c \leqslant n+1$ and $L_{N,m} * (\lceil m/2\rceil - 1) + 1 \leqslant W_{N,m} \leqslant L_{N,m} * (m-1)$, the values of s and $LL_{N,m}$ are uniquely determined, namely,

$$s = \lceil W_{N,m}/L_{N,m} \rceil \quad \text{and} \quad LL_{N,m} = W_{N,m} - (s-1) * L_{N,m}.$$

Note that $\lceil m/2\rceil \leqslant s \leqslant m-1$ and $LL_{N,m} \geqslant 1$. Therefore, the lemma follows. □

LEMMA 2. *For* $N > m$, $1 < \log_{m/\lceil m/2\rceil}(((N+1)*m)/((r+1)*\lceil m/2\rceil^{n+1})) \leqslant n+1$.

Proof. Since $r = \lceil (N+1)/m^n - 1\rceil$, we have $r \geqslant (N+1)/m^n - 1$. It is not difficult to prove that

$$\log_{m/\lceil m/2\rceil}\bigl(((N+1)*m)/\bigl((r+1)*\lceil m/2\rceil^{n+1}\bigr)\bigr) \leqslant n+1.$$

On the other hand, to prove $1 < \log_{m/\lceil m/2\rceil}(((N+1)*m)/((r+1)*\lceil m/2\rceil^{n+1}))$ for $N > m$ is equivalent to proving

$$m/\lceil m/2\rceil < ((N+1)*m)/\bigl((r+1)*\lceil m/2\rceil^{n+1}\bigr)$$

for $N > m$.

In the following we consider three cases, $m < N \leqslant 2*m-1$, $2*m \leqslant N \leqslant m^2-1$, and $m^2 \leqslant N$, to complete the proof.

CASE 1 ($m < N \leqslant 2*m-1$). In this case, $n=1$ and $r=1$. We have

$$\begin{aligned}&((m+1)*m)/\bigl((r+1)*\lceil m/2\rceil^{n+1}\bigr)\\ &\qquad < ((N+1)*m)/\bigl((r+1)*\lceil m/2\rceil^{n+1}\bigr)\end{aligned} \tag{1}$$

for $N > m$. The left-hand side of (1) equals

$$((m+1)*m)/\bigl(2*\lceil m/2\rceil^2\bigr) \geqslant m/\lceil m/2\rceil.$$

CASE 2 ($2*m \leqslant N \leqslant m^2-1$). In this case, $n=1$ and $2 \leqslant r = \lceil (N+1)/m - 1\rceil \leqslant m-1$. Moreover, since $r-1 < (N+1)/m - 1$, we have $r*m-1 < N$. Therefore,

$$(r/(r+1))*(m/\lceil m/2\rceil)^{n+1} < ((N+1)*m)/\bigl((r+1)*\lceil m/2\rceil^{n+1}\bigr) \tag{2}$$

can be derived. The left-hand side of (2) equals

$$\begin{aligned}&(r/(r+1))*(m/\lceil m/2\rceil)^2\\ &\quad \geqslant (2/3)*(m/\lceil m/2\rceil)*(m/\lceil m/2\rceil)\\ &\quad \geqslant (2/3)*(3/2)*(m/\lceil m/2\rceil) \quad (\text{since } m \geqslant 3,\ m/\lceil m/2\rceil \geqslant 3/2)\\ &\quad \geqslant m/\lceil m/2\rceil.\end{aligned}$$

CASE 3 ($N \geqslant m^2$). In this case, $n \geqslant 2$ and $1 \leqslant r = \lceil (N+1)/m^n - 1 \rceil \leqslant m-1$. Moreover, since $r - 1 < (N+1)/m^n - 1$, we have $r * m^n - 1 < N$. Therefore,

$$(r/(r+1)) * (m/\lceil m/2 \rceil)^{n+1} < ((N+1)*m)/\left((r+1)*\lceil m/2 \rceil^{n+1}\right) \quad (3)$$

can be derived. The left-hand side of (3) equals

$$(r/(r+1)) * (m/\lceil m/2 \rceil)^n * (m/\lceil m/2 \rceil) \geqslant (1/2)*(3/2)^n * (m/\lceil m/2 \rceil)$$
$$\geqslant m/\lceil m/2 \rceil. \qquad \square$$

THEOREM 1. *For an arbitrary $N > m$, the B-tree $T_{N,m}$ uniquely exists.*

Proof. From Lemma 1, we know that $T_{N,m}$ is uniquely specified by a pair of c and $W_{N,m}$, because the height of $T_{N,m}$ is $n = \lceil \log_m(N+1) \rceil - 1$ and the number of keys that are contained in the root is $r = \lceil (N+1)/m^n - 1 \rceil$. Therefore, to complete the proof, we show that for an arbitrary N, there exists a unique pair of c and $W_{N,m}$ satisfying the following constraints:

$$\begin{aligned} N &= r + (r+1)*(m-1)*(1+m+m^2+\cdots+m^{c-3}) \\ &\quad + (r+1)*m^{c-2}*(\lceil m/2 \rceil - 1) \\ &\quad * \left(1 + \lceil m/2 \rceil + \lceil m/2 \rceil^2 + \cdots + \lceil m/2 \rceil^{n-c}\right) + W_{N,m} \\ &= (r+1)*m^{c-2}*\lceil m/2 \rceil^{n-c+1} - 1 + W_{N,m}, \end{aligned} \quad (4)$$

$$\begin{aligned} &(r+1)*m^{c-2}*\lceil m/2 \rceil^{n-c+1}*(\lceil m/2 \rceil - 1) + 1 \\ &\quad \leqslant W_{N,m} \leqslant (r+1)*m^{c-2}*\lceil m/2 \rceil^{n-c+1}*(m-1), \end{aligned} \quad (5)$$

$$1 < c \leqslant n+1. \quad (6)$$

Equation (4) means that the total number of keys contained in $T_{N,m}$ equals N. Combining (4) and (5), we have

$$(r+1)*m^{c-2}*\lceil m/2 \rceil^{n-c+2} \leqslant N \leqslant (r+1)*m^{c-1}*\lceil m/2 \rceil^{n-c+1} - 1. \quad (7)$$

Inequality (7) is equivalent to

$$(r+1)*m^{c-2}*\lceil m/2\rceil^{n-c+2}-1<N\leqslant(r+1)*m^{c-1}*\lceil m/2\rceil^{n-c+1}-1,$$

from which

$$c-1<\log_{m/\lceil m/2\rceil}\big(((N+1)*m)/\big((r+1)*\lceil m/2\rceil^{n+1}\big)\big)\leqslant c \qquad (8)$$

can be derived.

Clearly, $\lceil\log_{m/\lceil m/2\rceil}(((N+1)*m)/((r+1)*\lceil m/2\rceil^{n+1}))\rceil$ is the unique solution of c that satisfies (8). By Lemma 2, the unique solution of c also satisfies (6).

On the other hand, the unique solution of $W_{N,m}$ can be obtained from (4) and the unique value of c. Moreover, it can be seen from (7) that the obtained value of $W_{N,m}$ satisfies (5). □

COROLLARY 1. *For the B-tree* $T_{N,m}$,

$$\begin{aligned}
n&=\lceil\log_m(N+1)\rceil-1,\\
r&=\lceil(N+1)/m^n-1\rceil,\\
c&=\Big\lceil\log_{m/\lceil m/2\rceil}\big(((N+1)*m)/\big((r+1)*\lceil m/2\rceil^{n+1}\big)\big)\Big\rceil,\\
V_{N,m}&=(r+1)*\big((m^{c-2}-1)/(m-1)+m^{c-2}\\
&\qquad *(\lceil m/2\rceil^{n-c+2}-1)/(\lceil m/2\rceil-1)\big)+1,\\
W_{N,m}&=N-(r+1)*m^{c-2}*\lceil m/2\rceil^{n-c+1}+1,\\
L_{N,m}&=(r+1)*m^{c-2}*\lceil m/2\rceil^{n-c+1},\\
s&=\lceil W_{N,m}/L_{N,m}\rceil,\\
LL_{N,m}&=W_{N,m}-(s-1)*L_{N,m}.
\end{aligned}$$

According to the construction of $T_{N,m}$, we have the following lemma.

LEMMA 3. *For each node* $\langle i,j\rangle$ *in* $T_{N,m}$:

(a) *When* $i=1$,

$$P_{\langle i,1\rangle,v}=\begin{cases}\langle 2,v\rangle, & \textit{for } 1\leqslant v\leqslant r+1,\\ \textit{undefined}, & \textit{for } r+1<v\leqslant m.\end{cases}$$

(b) *When* $1 < i < c$,

$$P_{\langle i,j\rangle,v} = \langle i+1, (j-1) * m + v\rangle, \quad \textit{for } 1 \leqslant v \leqslant m.$$

(c) *When* $c \leqslant i < n+1$,

$$P_{\langle i,j\rangle,v} = \begin{cases} \langle i+1, (j-1) * \lceil m/2\rceil + v\rangle, & \textit{for } 1 \leqslant v \leqslant \lceil m/2\rceil, \\ \textit{undefined}, & \textit{for } \lceil m/2\rceil < v \leqslant m. \end{cases}$$

(d) *When* $i = n+1$,

$$P_{\langle i,j\rangle,v} = \textit{undefined}, \quad \textit{for } 1 \leqslant v \leqslant m.$$

Let us consider a node $\langle i,j\rangle$ in $T_{N,m}$. If $1 < i < c$, the set of leaf nodes that precede the key $K_{\langle i,j\rangle,v}$ in in-order traversal consists of the leaf nodes of the subtrees with roots at $\langle i+1,1\rangle, \langle i+1,2\rangle, \ldots, \langle i+1,(j-1) * m+v\rangle$, respectively. If $i=1$ or $c \leqslant i \leqslant n+1$, the set of leaf nodes that precede $K_{\langle i,j\rangle,v}$ in in-order traversal can be identified similarly. Therefore, for each node $\langle i,j\rangle$ in $T_{N,m}$, $PREC_{N,m}(K_{\langle i,j\rangle,v})$ can be computed according to the following lemma.

LEMMA 4.

$$PREC_{N,m}(K_{\langle i,j\rangle,v})$$

$$= \begin{cases} v * m^{c-2} * \lceil m/2\rceil^{n-c+1}, \\ \quad \textit{if } i=1 \textit{ and } 1 \leqslant v \leqslant r, \\ ((j-1) * m + v) * m^{c-i-1} * \lceil m/2\rceil^{n-c+1}, \\ \quad \textit{if } 1 < i < c \textit{ and } 1 \leqslant v \leqslant m-1, \\ ((j-1) * \lceil m/2\rceil + v) * \lceil m/2\rceil^{n-i}, \\ \quad \textit{if } c \leqslant i < n+1 \textit{ and } 1 \leqslant v \leqslant \lceil m/2\rceil - 1, \\ j-1, \\ \quad \textit{if } i=n+1,\ 1 \leqslant j \leqslant LL_{N,m}, \textit{ and } 1 \leqslant v \leqslant s, \\ j-1, \\ \quad \textit{if } i=n+1,\ LL_{N,m} < j, \textit{ and } 1 \leqslant v \leqslant s-1, \\ \textit{undefined} \\ \quad \textit{otherwise}. \end{cases}$$

To facilitate the subsequent discussion, we introduce the tree $T_{N',m}$ which is obtained from $T_{N,m}$ by inserting one key into each of the leaf nodes that own only $s-1$ keys. Clearly, $T_{N',m}$ is the unique tree to be constructed for $N' = N + L_{N,m} - LL_{N,m}$. Note that $T_{N',m}$ and $T_{N,m}$ are of the same shape but different in that every leaf node of the former contains s keys but only the leftmost $LL_{N,m}$ leaf nodes of the latter contain s keys. The tree $T_{N',m}$ is helpful to the computation of $RANK_{N,m}(K_{\langle i,j\rangle,v})$. In the following, we introduce some properties about $T_{N',m}$.

LEMMA 5. *In $T_{N',m}$, the subtree whose root is at node $\langle i,j\rangle$ contains $(s+1)*m^{c-i}*\lceil m/2\rceil^{n-c+1}-1$ keys if $2 \leqslant i < c$ and $(s+1)*\lceil m/2\rceil^{n-i+1}-1$ keys if $c \leqslant i \leqslant n+1$.*

Proof. When $2 \leqslant i < c$, the number of keys that are contained in the subtree whose root is at node $\langle i,j\rangle$ is computed as the sum of $(m-1)*(1+m+m^2+\cdots+m^{c-i-1}) + (\lceil m/2\rceil - 1)*m^{c-i}*(1+\lceil m/2\rceil+\lceil m/2\rceil^2+\cdots+\lceil m/2\rceil^{n-c}) + s*m^{c-i}*\lceil m/2\rceil^{n-c+1}$, which can be simplified to $(s+1)*m^{c-i}*\lceil m/2\rceil^{n-c+1}-1$. When $c \leqslant i \leqslant n+1$, the computation is similar. □

Since the set of keys that precede $K_{\langle i,1\rangle,1}$ in in-order traversal of $T_{N',m}$ is exactly the set of keys that are contained in the subtree whose root is at node $\langle i+1,1\rangle$, the following lemma can be derived from Lemma 5.

LEMMA 6.

$$RANK_{N',m}(K_{\langle i,1\rangle,1}) = \begin{cases} (s+1)*m^{c-i-1}*\lceil m/2\rceil^{n-c+1}, & \text{if } 1 \leqslant i < c,\\ (s+1)*\lceil m/2\rceil^{n-i}, & \text{if } c \leqslant i < n+1,\\ 1, & \text{if } i = n+1. \end{cases}$$

Further, we have the following lemma.

LEMMA 7.

$$RANK_{N',m}(K_{\langle i,j+1\rangle,1}) = \begin{cases} RANK_{N',m}(K_{\langle i,j\rangle,1}) + (s+1)*m^{c-i}*\lceil m/2\rceil^{n-c+1}, \\ \qquad \text{if } 2 \leqslant i < c,\\ RANK_{N',m}(K_{\langle i,j\rangle,1}) + (s+1)*\lceil m/2\rceil^{n-i+1}, \\ \qquad \text{if } c \leqslant i < n+1,\\ RANK_{N',m}(K_{\langle i,j\rangle,1}) + s + 1, \\ \qquad \text{if } i = n+1. \end{cases}$$

Proof. The keys that are between $K_{\langle i,j\rangle,1}$ and $K_{\langle i,j+1\rangle,1}$, while traversing $T_{N',m}$ in in-order traversal, include the following four parts:

1. The keys in the subtree whose root is at $P_{\langle i,j\rangle,2}$.
2. The key $K_{\langle i,j\rangle,v}$ and the keys in the subtree whose root is at $P_{\langle i,j\rangle,v+1}$, where $2 \leqslant v \leqslant m-1$ if $2 \leqslant i < c$, $2 \leqslant v \leqslant \lceil m/2 \rceil - 1$ if $c \leqslant i < n+1$, and $2 \leqslant v \leqslant s$ if $i = n+1$.
3. The split key of nodes $\langle i,j\rangle$ and $\langle i,j+1\rangle$.
4. The keys in the subtree whose root is at $P_{\langle i,j+1\rangle,1}$.

By counting the above keys with the aid of Lemmas 3 and 5, the lemma follows. □

Similarly, since the keys that are between $K_{\langle i,j\rangle,v}$ and $K_{\langle i,j\rangle,v+1}$, while traversing $T_{N',m}$ in in-order traversal, are contained in the subtree whose root is at $P_{\langle i,j\rangle,v+1}$, we have the following lemma.

LEMMA 8.

$$RANK_{N',m}(K_{\langle i,j\rangle,v+1})$$

$$= \begin{cases} RANK_{N',m}(K_{\langle i,j\rangle,v}) + (s+1) * m^{c-i-1} * \lceil m/2 \rceil^{n-c+1}, \\ \quad \textit{if } i = 1 \textit{ and } 1 \leqslant v < r, \\ RANK_{N',m}(K_{\langle i,j\rangle,v}) + (s+1) * m^{c-i-1} * \lceil m/2 \rceil^{n-c+1}, \\ \quad \textit{if } 1 < i < c \textit{ and } 1 \leqslant v < m-1, \\ RANK_{N',m}(K_{\langle i,j\rangle,v}) + (s+1) * \lceil m/2 \rceil^{n-i}, \\ \quad \textit{if } c \leqslant i < n+1 \textit{ and } 1 \leqslant v < \lceil m/2 \rceil - 1, \\ RANK_{N',m}(K_{\langle i,j\rangle,v}) + 1, \\ \quad \textit{if } i = n+1,\ 1 \leqslant j \leqslant LL_{N,m}, \textit{ and } 1 \leqslant v < s, \\ RANK_{N',m}(K_{\langle i,j\rangle,v}) + 1, \\ \quad \textit{if } i = n+1,\ LL_{N,m} < j, \textit{ and } 1 \leqslant v < s-1, \\ \textit{undefined}, \\ \quad \textit{otherwise}. \end{cases}$$

According to Lemmas 6, 7, and 8, $RANK_{N',m}(K_{\langle i,j\rangle,v})$ can be computed as follows. First, by Lemma 6, $RANK_{N',m}(K_{\langle i,1\rangle,1})$ is computed. Then, by applying Lemma 7 repeatedly, $RANK_{N',m}(K_{\langle i,j\rangle,1})$ is computed. Finally, by applying Lemma 8 repeatedly, $RANK_{N',m}(K_{\langle i,j\rangle,v})$ is computed. Therefore, we have the following theorem.

THEOREM 2.

$$RANK_{N',m}(K_{\langle i,j\rangle,v}) = \begin{cases} v*(s+1)*m^{c-i-1}*\lceil m/2\rceil^{n-c+1}, & \text{if } i=1 \text{ and } 1\leqslant v\leqslant r,\\ ((j-1)*m+v)*(s+1)*m^{c-i-1}*\lceil m/2\rceil^{n-c+1}, & \text{if } 1<i<c \text{ and } 1\leqslant v\leqslant m-1,\\ ((j-1)*\lceil m/2\rceil+v)*(s+1)*\lceil m/2\rceil^{n-i}, & \text{if } c\leqslant i<n+1 \text{ and } 1\leqslant v\leqslant\lceil m/2\rceil-1,\\ (j-1)*(s+1)+v, & \text{if } i=n+1,\ 1\leqslant j\leqslant LL_{N,m}, \text{ and } 1\leqslant v\leqslant s,\\ (j-1)*(s+1)+v, & \text{if } i=n+1,\ LL_{N,m}<j, \text{ and } 1\leqslant v\leqslant s-1,\\ \text{undefined}, & \text{otherwise}. \end{cases}$$

With the aid of Theorem 2, $RANK_{N,m}(K_{\langle i,j\rangle,v})$ can be computed easily. Note that only the leftmost $LL_{N,m}$ leaf nodes of $T_{N,m}$ own s keys, whereas every leaf node of $T_{N',m}$ owns s keys. Therefore, for each node $\langle i,j\rangle$, $RANK_{N,m}(K_{\langle i,j\rangle,v}) = RANK_{N',m}(K_{\langle i,j\rangle,v})$ if $PREC_{N,m}(K_{\langle i,j\rangle,v}) \leqslant LL_{N,m}$ and $RANK_{N,m}(K_{\langle i,j\rangle,v}) = RANK_{N',m}(K_{\langle i,j\rangle,v}) - (PREC_{N,m}(K_{\langle i,j\rangle,v}) - LL_{N,m})$ if $PREC_{N,m}(K_{\langle i,j\rangle,v}) > LL_{N,m}$. We summarize the result in the following theorem.

THEOREM 3. $RANK_{N,m}(K_{\langle i,j\rangle,v}) = RANK_{N',m}(K_{\langle i,j\rangle,v}) - max\{0, PREC_{N,m}(K_{\langle i,j\rangle,v}) - LL_{N,m}\}$.

According to Theorem 3, $RANK_{N,m}(K_{\langle i,j\rangle,v})$ can be computed after $LL_{N,m}$, $PREC_{N,m}(K_{\langle i,j\rangle,v})$, and $RANK_{N',m}(K_{\langle i,j\rangle,v})$ have been computed, respectively, according to Corollary 1, Lemma 4, and Theorem 2.

4. PARALLEL CONSTRUCTION ALGORITHMS

A convenient data structure to represent a search tree is a linear array. In this section, we assume that $T_{N,m}$ is stored in a linear array of length $V_{N,m}$. Therefore, before presenting the parallel construction algorithms, we need to specify a linear ordering for the nodes of $T_{N,m}$ such that the node with order k, $1 \leqslant k \leqslant V_{N,m}$, is represented by the kth element of the linear array. A simple approach to do so is to number the nodes according to their breadth-first search order. For example, Table 1 shows the specified linear ordering for the nodes of the B-tree that was depicted in Figure 1a. For each node $\langle i,j\rangle$ in $T_{N,m}$, it is not difficult to determine its order in the specified linear ordering. Let $ORDER_{N,m}(\langle i,j\rangle)$ denote the order of node $\langle i,j\rangle$ in $T_{N,m}$. We can determine $ORDER_{N,m}(\langle i,j\rangle)$ by the following lemma.

LEMMA 9. *Let $\langle i,j\rangle$ be a node of $T_{N,m}$. Then,*

$$ORDER_{N,m}(\langle i,j\rangle)=\begin{cases}1, \\ \quad \text{if } i=1, \\ 1+(r+1)*(m^{i-2}-1)/(m-1)+j, \\ \quad \text{if } 2\leqslant i\leqslant c, \\ 1+(r+1)*(m^{c-2}-1)/(m-1)+(r+1)*m^{c-2} \\ \quad *(\lceil m/2\rceil^{i-c}-1)/(\lceil m/2\rceil-1)+j, \\ \quad \text{if } c<i\leqslant n+1.\end{cases}$$

On the other hand, given the above specified order, we can also determine the corresponding node. Let $NODE_{N,m}(k)$ denote the node in $T_{N,m}$ whose order is k. Clearly, $NODE_{N,m}(k)=ORDER_{N,m}^{-1}(k)$. We can determine $NODE_{N,m}(k)$ by the following lemma.

LEMMA 10. *For a given order k, $1\leqslant k\leqslant V_{N,m}$, if $NODE_{N,m}(k)=\langle i,j\rangle$, then*:

(a) *When $k=1$,*

$$i=1 \quad and \quad j=1.$$

(b) *When $1<k\leqslant 1+(r+1)*(m^{c-1}-1)/(m-1)$,*

$$i=\lceil \log_m((k-1)*(m-1)/(r+1)+1)\rceil+1$$

TABLE 1

The Specified Linear Ordering for the Nodes of the B-tree that was Depicted in Figure 1a

Node $\langle i,j\rangle$	$\langle 1,1\rangle$	$\langle 2,1\rangle$	$\langle 2,2\rangle$	$\langle 3,1\rangle$	$\langle 3,2\rangle$	$\langle 3,3\rangle$	$\langle 3,4\rangle$	$\langle 3,5\rangle$	$\langle 3,6\rangle$	$\langle 4,1\rangle$	$\langle 4,2\rangle$
Order k	1	2	3	4	5	6	7	8	9	10	11
Node $\langle i,j\rangle$	$\langle 4,3\rangle$	$\langle 4,4\rangle$	$\langle 4,5\rangle$	$\langle 4,6\rangle$	$\langle 4,7\rangle$	$\langle 4,8\rangle$	$\langle 4,9\rangle$	$\langle 4,10\rangle$	$\langle 4,11\rangle$	$\langle 4,12\rangle$	
Order k	12	13	14	15	16	17	18	19	20	21	

and

$$j = k - \left(1 + (r+1) * (m^{i-2} - 1)/(m-1)\right).$$

(c) *When* $1 + (r+1) * (m^{c-1} - 1)/(m-1) < k \leqslant V_{N,m}$,

$$i = \Big\lceil \log_{\lceil m/2 \rceil}\big((k - (1 + (r+1) * (m^{c-2} - 1)/(m-1)))$$
$$* (\lceil m/2 \rceil - 1)/((r+1) * m^{c-2}) + 1\big)\Big\rceil + c - 1$$

and

$$j = k - \Big(1 + (r+1) * (m^{c-2} - 1)/(m-1) + (r+1) * m^{c-2}$$
$$* \left(\lceil m/2 \rceil^{i-c} - 1\right)/(\lceil m/2 \rceil - 1)\Big).$$

Since each node of $T_{N,m}$ is uniquely represented by an element of a linear array of length $V_{N,m}$, for each element of the linear array, we have to determine which node of $T_{N,m}$ it represents, which keys in the given sorted list it will keep, and which elements of the linear array are its children, in order to construct $T_{N,m}$. To say more concretely, for the kth element of the linear array, $1 \leqslant k \leqslant V_{N,m}$, we have to determine $NODE_{N,m}(k)$, $RANK_{N,m}(K_{\langle i,j \rangle, v})$, $1 \leqslant v \leqslant m-1$, and $ORDER_{N,m}(P_{\langle i,j \rangle, u})$, $1 \leqslant u \leqslant m$, where $\langle i,j \rangle = NODE_{N,m}(k)$. Since the above computations can be performed independently for all the elements of the linear array, we can derive parallel construction algorithms, running on the shared-memory SIMD computer, in which all the elements are processed in parallel. The parallel construction algorithms consist of the following two steps:

Step 1. Compute n, r, c, $V_{N,m}$, $W_{N,m}$, $L_{N,m}$, s, and $LL_{N,m}$ according to Corollary 1.

Step 2. Compute $NODE_{N,m}(k)$, $RANK_{N,m}(K_{\langle i,j \rangle, v})$, $1 \leqslant v \leqslant m-1$, and $ORDER_{N,m}(P_{\langle i,j \rangle, u})$, $1 \leqslant u \leqslant m$, where $\langle i,j \rangle = NODE_{N,m}(k)$, according to Lemmas 3 and 4, Theorems 2 and 3, and Lemmas 9 and 10 for all k, $1 \leqslant k \leqslant V_{N,m}$.

The time complexity of the parallel construction algorithms are dependent on how fast the set S of values m^n, $\lceil m/2 \rceil^{n+1}$, m^{c-2}, $\lceil m/2 \rceil^{n-c+2}$, $\lceil m/2 \rceil^{n-c+1}$, m^{c-i-1}, $\lceil m/2 \rceil^{n-i}$, m^{c-i}, $\lceil m/2 \rceil^{n-i+1}$, m^{i-2}, $\lceil m/2 \rceil^{i-c+1}$, $\lceil m/2 \rceil^{i-c}$, and m^{c-1} can be computed. In the set S, m is given and the values of n and c are fixed when N is given, but the value of i varies with the level of the node being considered. That is, $i = i_k$, when node $\langle i_k, j_k \rangle$ is being processed. Sequentially, $O(\log n)$ time is required to compute the set S. However, if N processors are used, the set S can be computed in $O(1)$ time (explained later). In the following, we present the parallel construction algorithms running on the EREW model and the CREW model, respectively.

PARALLEL CONSTRUCTION ALGORITHM ON THE EREW MODEL

A simple parallel algorithm for constructing $T_{N,m}$ on the EREW model is to let all processors P_k, $1 \leqslant k \leqslant V_{N,m}$, process the kth element simultaneously. The time complexity is $O(\log\log N)$ $[=O(\log n)]$, which is the time requirement for P_k to compute the set S for $i = i_k$, where $\langle i_k, j_k \rangle = NODE_{N,m}(k)$. Since $V_{N,m} \leqslant N$ processors are needed, the parallel algorithm is not cost-optimal. According to the following fact, we can propose a cost-optimal parallel algorithm, where $V_{N,m}/\log\log N$ processors are used, while $O(\log\log N)$ time is retained.

FACT 1. Let $\langle i_k, j_k \rangle = NODE_{N,m}(k)$ and $\langle i_{k+1}, j_{k+1} \rangle = NODE_{N,m}(k+1)$. Then, $i_{k+1} - i_k = 0$ or 1. That is, $NODE_{N,m}(k)$ and $NODE_{N,m}(k+1)$ are two nodes at the same level or at adjacent levels.

As a result of Fact 1, when we have obtained the set S of values for $i = i_k$, we can obtain the set S of values for $i = i_{k+1}$ in additional constant time, where $\langle i_k, j_k \rangle = NODE_{N,m}(k)$ and $\langle i_{k+1}, j_{k+1} \rangle = NODE_{N,m}(k+1)$. This means that the $(k+1)$th element can be processed in constant time after the kth element has been processed. The cost-optimal parallel algorithm is simply to let each processor process $\log\log N$ consecutive elements. Each processor will process the assigned elements in increasing order of their indices and therefore it will take $O(\log\log N)$ time for the first one and $O(1)$ time for each of the others. Thus, totally $O(\log\log N)$ time is required for each processor.

THEOREM 4. *The B-tree $T_{N,m}$ can be constructed in $O(\log\log N)$ time on an EREW shared-memory SIMD computer with $V_{N,m}/\log\log N \leqslant N/\log\log N$ processors, which is cost-optimal.*

PARALLEL CONSTRUCTION ALGORITHM ON THE CREW MODEL

From the above discussion, we know that we have to speed up the computation of the set S in order to obtain a faster parallel construction algorithm. Fortunately, we have an approach to compute the set S for all i's, $1 \leqslant i \leqslant n+1$, in $O(1)$ time. As a result, an $O(1)$ time parallel construction algorithm is derived. In this approach, N processors are used to fill in two tables $POWER_{\lceil m/2 \rceil}[1 \cdots n+1]$ and $POWER_m[1 \cdots n+1]$, where the contents of $POWER_{\lceil m/2 \rceil}[i]$ and $POWER_m[i]$, $1 \leqslant i \leqslant n+1$, are the values of $\lceil m/2 \rceil^i$ and m^i, respectively. Here we only describe the approach to fill in the table $POWER_m[1 \cdots n+1]$. A similar approach to fill in the table $POWER_{\lceil m/2 \rceil}[1 \cdots n+1]$ easily can be derived. In the approach, each processor P_k, $1 < k \leqslant N$, is first to compute $u_k = \lceil \log_m k \rceil$ and $l_k = \lfloor \log_m k \rfloor$. If $u_k = l_k$, P_k fills $POWER_m[u_k]$ with the value of k. Then (since $m^n \leqslant N \leqslant m^{n+1} - 1$), processor P_N fills $POWER_m[n+1]$ with the value of $m * POWER_m[n]$. After the two tables have been established, $T_{N,m}$ can be constructed in $O(1)$ time by letting each processor P_k, $1 \leqslant k \leqslant V_{N,m}$, process the kth element simultaneously. Now the set S of values for any i can be obtained in $O(1)$ time by looking up the tables.

THEOREM 5. *The B-tree $T_{N,m}$ can be constructed in $O(1)$ time on the CREW shared-memory SIMD computer with N processors, which is cost-optimal.*

5. CONCLUDING REMARKS

Parallel algorithms for constructing binary and m-way search trees have been proposed in the literature [7, 9, 10]. In this paper, we have proposed two parallel algorithms for constructing a B-tree of order m. The main advantage of the B-tree over the m-way search tree, i.e., guaranteeing logarithmic access time even when it is dynamically changed, comes from the flexibility of its structure. However, the flexibility also causes the difficulty of deriving parallel algorithms for constructing it. The two parallel algorithms that we have proposed in this paper were designed on the shared-memory SIMD computer: one, based on the EREW model, uses $N/\log\log N$ processors and requires $O(\log\log N)$ time; the other, based on the CREW model, uses N processors and requires $O(1)$ time. Both parallel algorithms are cost-optimal. If the number of available processors is fixed, say p, it is not difficult to see that the proposed algorithms can run in $O(N/p)$ time, which is also cost-optimal.

There have been many variants of B-trees, such as B*-trees, B^+-trees, and prefix B^+-trees [5]. The parallel algorithms we have proposed in this paper cannot be used to construct them. It is still open for the interested readers to discover if there exist efficient parallel algorithms for constructing them.

REFERENCES

1. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.
2. R. Bayer and E. M. McCreight, Organization and maintenance of large ordered indices. *Acta Inform.* 1:173–189 (1972).

3. M. R. Brown and R. E. Tarjan, Design and analysis of data structure for representing sorted lists. *SIAM J. Comput..* 9(3):594–614 (1980).
4. H. Chang and S. S. Iyengar, Efficient algorithm to globally balance binary search trees. *Commun. ACM* 27(7):695–702 (1984).
5. D. Comer, The ubiquitous B-tree. *Comput. Surveys* 11:121–137 (1979).
6. A. C. Day, Balancing a binary tree. *Comput. J.* 19:360–361 (1976).
7, E. Dekel, S. Peng, and S. S. Iyengar, Optimal parallel algorithms for constructing and maintaining a balanced *m*-way search tree. *Int. J. Parallel Program.* 15(6):503–528 (1986).
8. D. E. Knuth, *The Art of Computer Programming*, Vol. 3: *Sorting and Searching.* Addison-Wesley, Reading, MA, 1973.
9. A. Moitra and S. S. Iyengar, A maximally parallel balancing algorithm for obtaining complete balanced binary trees. *IEEE Trans. Comput.* C-34(6):563–565 (1985).
10. A. Moitra and S. S. Iyengar, Derivation of a parallel algorithm for balanced binary trees. *IEEE Trans. Software Eng.* SE-12(3):442–449 (1986).
11. B. F. Wang and G. H. Chen, Cost-optimal parallel algorithms for constructing 2-3 trees. *J. Parallel Distributed Comput.* 11(3):257–261 (1991).

Received 2 December 1991; revised 8 October 1992