J.King Saud Univ.,Vol.8 , *Computer & Information Sci.*,pp.111-138 (A.H 1416/1996)

Time Cost Analysis of Parallel Structures with Multi-Communication Nodes in a Shared Memory Environment

Reda A. Ammar

U-155, Computer Science and Engineering Dept., University of Connecticut
Storrs, Connecticut 06269-3155,USA

(Received, 07 June. 1993; accepted for publication, 20 June 1994)

Abstract. In the paper the time costs of several parallel computation structures are analyzed. These analyses are based on the assumptions that the processes communicate implicitly via the shared memory and that a locking mechanism is imposed on the access to shared variables. In previous work, an approach to estimating a set of special parallel computation structures has been developed. In this paper, we expand this approach and propose a new technique for covering more general parallel computation structures.

1. Introduction

Time cost is a very important factor in determining the performance of either parallel or sequential software. However, for a parallel computation (the term "parallel computation" is used when we talk about the code level rather than the system level [1,2], it is more difficult to evaluate the time cost than that of a sequential computation. The reasons for this are two-fold. First, a parallel computation structure is more complicated than a sequential one since a parallel structure constains more than one paths executed at the same time. Second, parallel computations require additional communication between different processes , which may not be present in a sequential

computation. Traditionally, there are two mechanisms used for the communication of a parallel compution: shared memory for implicit communication and message passing for explicit communication. Our major goal is to evaluate the time cost of a parallel computation using the shared memory mechanism for communication.

Our approach is based on the computation structure model [1], that can model the detailed time cost of a sequential or parallel computation [3,4]. Since the model can not represent the communication behavior in a parallel computation, two new types of nodes, a *lock node* and *an unlock node*, are added to the computation structure model [5]. The *lock node* is used to obtain locks on shared data and the *unlock* node is used to release locks. The purpose of the locks is to control access to the shared message areas so that a read (write) does not occur until a write (read) has completed. Based on this modified model, the time costs of a set of special cases of parallel computation structures are derived [1,2,6]. In this paper, we expand our investigation to evaluate analytically the time cost of a given parallel structure when some or all parallel branches have two conflicting communication nodes. We assume that is not possible for the second communication node to proceed before finish executing the first communication nodes in all branches.

As the number of communication nodes in each branch increases beyond the stated limit, the analytical approach becomes extremely difficult and should be guided by simulation [7,8]. In our approach, we assume that the system is dedicated to execute the given parallel structure.

This paper is organized into three sections and a conclusion. In section 2, we review some background information and two parallel computation structures [6] that are used as a basis of our work in this paper. In Section 3, four cases of parallel computation structures are discussed and the time costs of these structures are derived. In Section 4, an example is used to illustrate the analysis of time cost for a real application. Finally, a short conclusion is given in Section 5.

2. Background Information

2.1. Communication Model

A computation structure model consists of two directed graphs, a *control flow graph* and a *data flow graph*. The control flow graph shows the order of operations in a computation while the data flow graph shows the relationship between operations and data.

A control flow graph contains a *start* node, an *end* node, *operation* nodes, *decision* nodes, or nodes, *fork* nodes, and *join* nodes. The start and end nodes indicate the beginning and end of a computation, respectively. An operation node represents an operation to be performed in a computation. A decision node is used at a branch point to check conditions. An or node serves as a junction point to merge different branches together. It works as the logic or except only one signal at most is expected to cross the or node at a given time. A fork node splits the execution path into a number of parallel execution paths [9]. A join node merges parallel execution paths into a single execution path. A computation is sequential if it does not contain any fork or join nodes, otherwise, it is parallel. In a computation model, it is assumed that there is an activation signal. A computation begins when an activation signal enters the start node. When a signal enters an operation node, the operation specificed by the node is performed and the signal then leaves the node. When a signal arrives at a decision node, the decision node checks some conditions and the signal leaves the node from one of its outgoing edges depending on the result of the checking. When a signal arrives at an or node (only one signal will arrive to the or node at a given time) from one of its incoming edges, the signal immediately leaves the node from its outgoing edge. When a signal reaches a fork node, the fork node creates parallel execution paths and an activation signal is put on each parallel path and all the branches start executing their operations simultaneously [1]. When a join node receives signals from all of its incoming edges, an activation signal is created and the signal leaves the join node from its an activation signal finally arrives at the *end* node, the execution terminates.

A data flow graph contains *operation* nodes and *data* nodes. An edge goes from a *data* node to an *operation* node if the corresponding data item is the input to the operation. Similarly, an edge goes from an operation node to a data node if the corresponding data item is the output to the operation.

The time cost of a computation is defined as follows. Each node in the control graph is associated with a time cost which is equal to the time that is required to perform the operations specified by the node (we have found that the execution time of an or node is very small and can be ignored [1]. When a node with time cost C receives all of the required activation signals, the execution of the specified operations are started and after C amount of time, activation signals leave the node. The time cost of a computation is then defined as the time for an activation signal to travel from the start node to the end node. Based on this model, analytic techniques have been developed to derive time costs of both sequential and parallel computations [3,4].

To model communication, a locking technique [5] is added to the computation structure model [1] to manipulate the accesses to shared data. It is assumed that each shared data item is associated with two locks, a read lock and a write lock. To read a shared data item, a read lock must be obtained on that shared data item. Similarly, a write lock on a shared data item is required if a write operation is to be performed. Several operations performed in parallel can read a shared data item at the same time. However, no other read or write lock on shared data item is granted if a write lock on that shared data item is being held.

To include data access control into the computation structure model, two new types of nodes, a lock node and an unlock node, are added to the model. The lock node is used to obtain locks on shared data and the unlock node is used to release locks. Each lock and unlock has an incoming edge and an outgoing edge associated with it in the control flow graph. A lock node is required to obtain a read lock on a data item X, if X is an input to the lock node in the data flow graph. Similarly, a lock node is required to obtain a write lock on X, if X is an output from the lock node in the data flow graph. If X is connected with an unlock node in the data flow graph, the unlock node can release a read or write lock on X, depending on whether X is an input or output of the unlock node. In Fig. 1, the lock node is required to obtain read locks on X and Z, and write lock on Y. The forms of lock and unlock nodes are consistent with the forms of other nodes in the computation structure model.

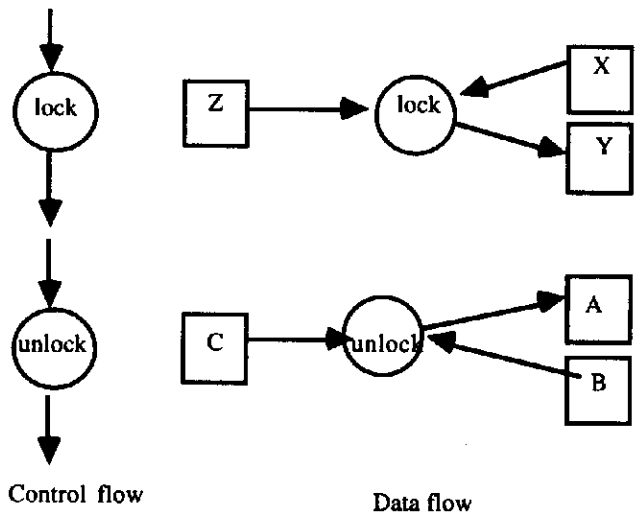

Fig. 1. Lock and unlock nodes

It is assumed that a *lock* node obtains its required locks at the same time. If a *lock* node cannot obtain all of the required locks, it is put into a lock *waiting queue*. The waiting lock nodes are checked in the order of their arrivals to see if their requests can be satisfied. Once all of the requests of a *lock* node are satisfied, the lock node is removed from the queue and is allowed to access the critical section without any interruption (i.e. non pre-emptive).

The time cost derived from the modified computation structure model is defined as the time for an activation signal to travel from the start node to the end node. When an activation signal enters an unlock node, the unlock node starts to release locks. The signal then leaves the node after all of the requested locks have been released. The time cost of an unlock node is equal to the time to release the requested locks. When an activation signal enters a lock node, the lock node must wait for the required locks. When all of the locks on required data are available, the lock node manipulates these locks. After all the required locks are obtained, the activation signal leaves the lock node. The time cost of a lock node is equal to the sum of the time to wait for the required locks and the time to manipulate the locks. The time cost of an unlock node and the amount of time a lock node manipulates locks can also be easily determined. Generally, they depend on the number of locks to be obtained or released, and the type of locks to be manipulated (whether read or write locks). The time cost of a lock node, however, is very difficult to analyze. This occurs since the waiting time cost of a lock node depends on other lock nodes in the computation.

To provide a framework of the discussion of this paper, we must introduce some basic terms and definitions. Given a *lock* node $lock_i$ define its write lock set, $W(lock_i)$, as the set of data on which $lock_i$ requires write locks; and define its read lock set, $R(lock_i)$, as the set of data on which $lock_i$ requires read locks. This leads to the following definitions :

Definition 1: Two lock nodes, $lock_i$ and $lock_j$, are in conflict if the following condition is satisfied:

$$W(lock_i) \cap W(lock_j) \cup W(lock_i) \cap R(lock_j) \cup R(lock_i) \cap W(lock_j) \neq \{\}$$

Definition 2: Two lock nodes are in parallel if they are in different paths of a parallel structure.

Definition 3: Two lock nodes are in independent of each other if they are not in

conflict; or if they are in conflict, but not in parallel.

Given the model defined so far, there are some uncertainties that may occur. Specifically, when more than one lock node in parallel requires conflict locks on a shared data item (i.e., read and write locks, or write and write locks). Consider the parallel structure in Fig. 2.

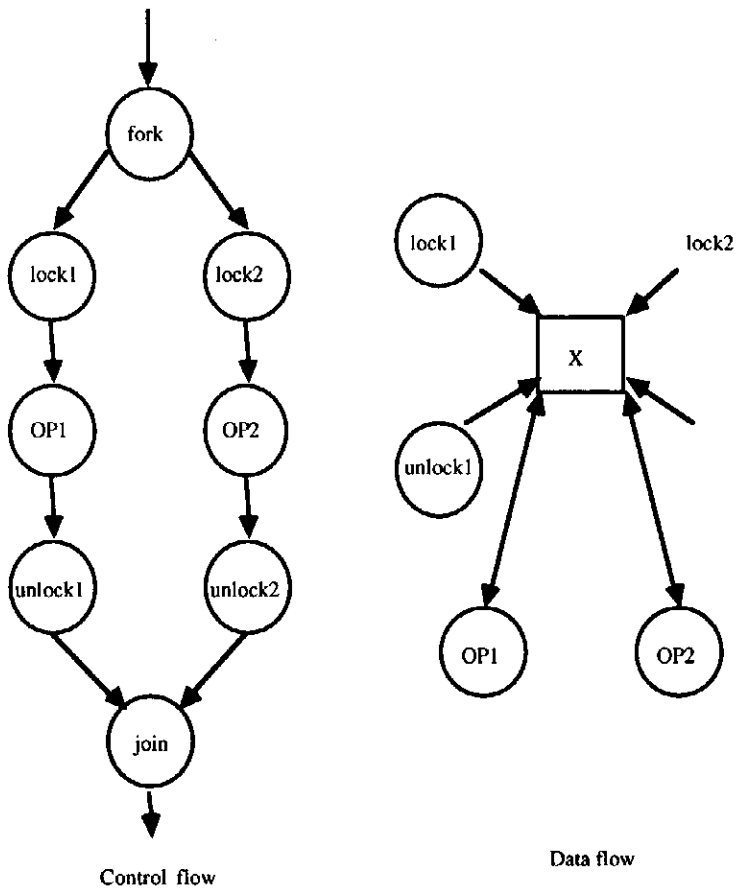

Fig. 2. Lock conflict condition

A conflict occurs in the parallel structure since $lock_1$ and $lock_2$ both require write locks on X at the same time. When this occurs, it is assumed that the computer system will arbitrarily choose one lock node to obtain its required locks first. The probability a lock node to obtain its required locks is assumed to be uniformly distributed. If any uncertainty occurs, the average time cost is used as the time cost of the computation.

We assume that all communications nodes of any parallel structure considered in this paper are in conflict. We do not use a specific scheduling policy or give different branches. Instead, we try all possible orderings of parallel branches, evaluate the execution time cost of each case, and then take the average of the results.

2.2. Previous Work

Based on the communication model defined, the analyses of a set of special cases of parallel computation structures are estimate [2,6]. All these parallel structures have lock-conflict conditions and have one lock node in each path. Two parallel structures are discussed as the starting point for the work presented in this paper.

2.2.1. Basic parallel computation structure 1

The basic parallel computation structure, taken from [6], is shown in Fig. 3. The time cost for this structure is computed as follows. First, assume that all of the op_{i1} nodes have the same time costs; any two lock nodes are in conflict; and a lock node is independent of any lock outside the parallel structure. Now let:

$$time\ cost\ of\ op_{i1} = C1_i$$
$$time\ cost\ of\ op_{i2} = C2_i$$
$$time\ cost\ of\ lock_{i1} = CL1_i$$
$$time\ cost\ of\ unlock_{i1} = CU1_i$$
$$time\ cost\ of\ fork = CF$$
$$time\ cost\ of\ join = CJ$$

then, the time cost of this parallel structure is equal to :

$$TC = CF + \frac{1}{n!} * \sum_{i=1}^{n!} \max_{1 \le j \le n} \left\{ C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + C3_{ij} \right\} + CJ \qquad (1)$$

where (i_1 , i_2 , ... , i_n) is the i'th permutation of (1,2, ... , n).

Since all of the op_{i1} nodes have the same time costs, the $lock_{i1}$ nodes will request the lock at the same time. Based on our assumptions, the probability of selecting a lock node when uncertainties occur is uniformly distributed. Since we are interested in estimating the expected time cost of parallel structures, we must consider all possible orders of $lock_{i1}$ nodes to obtain locks. The n! in equation (1) is the total number of the possible permutations of the n lock nodes. Expression (2) within the max condition is the time cost of a specific path with the summation representing the

waiting time for the lock node to obtain the locks.

$$C1_{ij}+\sum_{k=1}^{j}(CL1_{ik}+C2_{ik}+CU1_{ik})+C3_{ij} \quad (2)$$

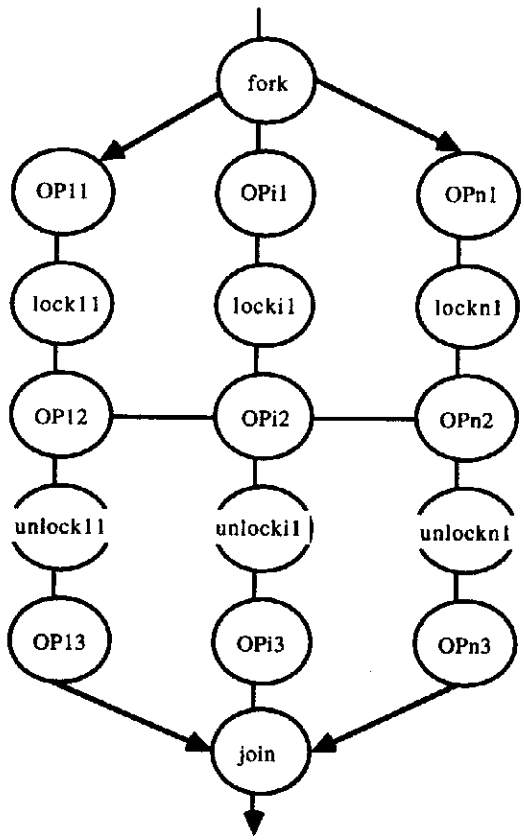

Fig.3.Parallel structure of *Opil*

2.2.2. Basic parallel computation structure 2

This parallel structure given in Fig. 3 has all of the properties of parallel structure 1, except that the time costs of all op_{il} nodes are different. Without loss of generality, assume that $C1_1 < ... < C1_i < ... C1_n$. Then, the time cost of this second parallel structure, as given in [2], is equal to :

$$TC = CF + \max_{i,j}\left\{C1_i+\sum_{k=i}^{j}(CL1_k+C2_k+CU1_k)+C3_j\right\}+CJ \quad (3)$$

Since all of the op_{il} nodes have different time costs, the $lock_{il}$ nodes will request

locks in a fixed order, i.e., $lock_{11}$ obtains the lock first, then $lock_{21}$, and so on. The expression within the max condition is the time cost of a specific path with the summation representing the waiting time for the lock node to obtain the locks.

3. Time Cost Analysis of Parallel computations

In [2,6], time analyses of a set of special parallel structures were presented. Since all of the parallel structures that were discussed only contain one lock node in each path, we want to investigate more general parallel structures which contain more than one lock nodes in some paths. In this section, we propose and discuss the time cost analyses of four parallel structures. We begin by presenting a simple example that demonstrates our approach. Then, for each of the proposed structures we present a brief intuitive explanation, a theorem and a proof. All of these strurctures consist of one or more paths containing two lock nodes, and we assume that any two lock nodes are in conflict and a lock node is independent of any lock node outside the parallel structure. In order to analyze a general parallel structure, we do the analysis of a prallel structure with one path containing two lock nodes first. Then we continue to analyze structure with two, and m paths containing two lock nodes.

Section parallel computation structure1 is the analysis of a parallel structure with a path containing two lock nodes. Section parallel computation structure2 is the analysis of a parallel structure with two path containing two lock nodes. Section parallel computation structure3 presents our general analysis of a parallel structure with m paths containing two lock locks. Section parallel computation structure4 present the analysis of a general parallel structure with different assumptions.

3.1. Our Approach

We begin by considering the parallel computation structure given in Fig. 4. In this structure, there are two paths executed in parallel with one path containing one lock and one path containing two locks.

In order to estimate the time cost of this computation, we must consider following cases:

* **Case 1:** Assume $lock_{21}$ conflicts with $lock_{11}$ and does not conflict with $lock_{12}$, and op_{11} and op_{21} have the same time costs. Based on these assumptions, this structure can be reduced to the basic parallel structure 1 in Figure 3, since $lock_{21}$ does not conflict with $lock_{12}$. Let op_{13} , be equal to $op_{13} + lock_{12} + op_{14}$

$+unlock_{12} + op_{15}$, then by applying equation (1), the time cost of this case is

$$TC = CF + \frac{1}{2} * \sum_{i=1}^{2} \max_{1 \le j \le 2} \left\{ C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + C3_{ij} \right\} + CJ \qquad (4)$$

* **Case 2:** Assume $lock_{21}$ conflicts with $lock_{12}$ and does not conflict with $lock_{11}$, and op_{11} and op_{21} have the same time costs. This case is similar to case 1, and can also be reduced to the basic parallel structure 1 in Figure 3. Let op_{11}, be equal to $op_{11} + lock_{11} + op_{12} + unlock_{11} + op_{13}$, then by applying equation (1), we can get the same time cost as equation (4).

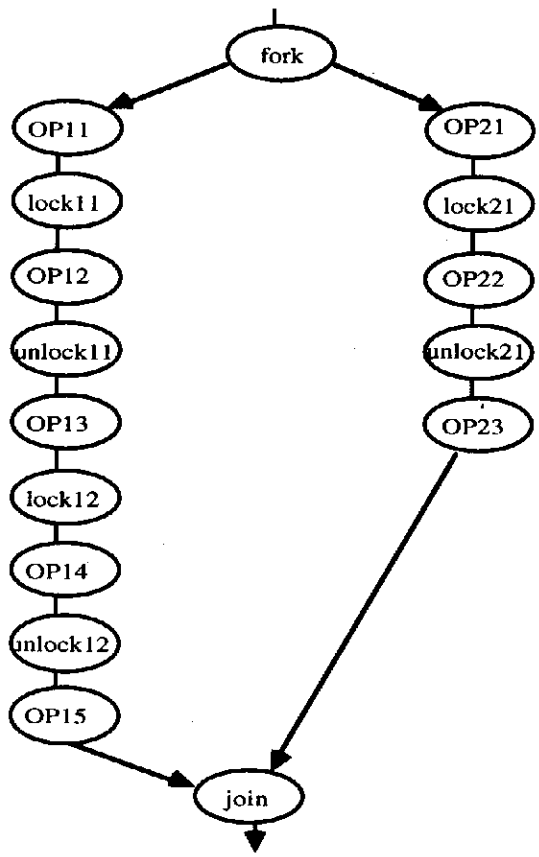

Fig. 4. An example of parallel computation

* **Case 3:** Assume lock11 conflicts with lock12 and does not conflict with lock21, and op11 and op21 have the same time costs. In this case, two execution paths are independent of each other, since lock21 does not conflict with either

lock11 or lock12 . So the time cost of this case equal to :

$$TC = CF + \max \{ C1_1 + CL1_1 + C2_1 + CU1_1 + C3_1 + CL2_1 + C4_1 + CU2_1 + C5_1 \ , C1_2 + CL1_2 + C2_2 + CU1_2 + C3_2 \ \} + \mathrm{CJ} .$$

* **Case 4:** Assume $lock_{11}$, $lock_{12}$ and $lock_{21}$ are in conflict with each other and op_{11} and op_{21} have the same time costs. In this case, we must consider two possible situations:

a) $lock_{21}$ otains the lock first.

In this situation, $lock_{11}$ must wait until $lock_{21}$ releases the lock, before it can obtain the lock, so the time cost is equal to :

$$TC = CF + \max \{ C1_2 + CL1_2 + C2_2 + CU1_2 + CL1_1 + C2_1 + CU1_1 + C3_1 + CL2_1 + C4_1 + CU2_1 + C5_1, \ C1_2 + CL1_2 + C2_2 + CU1_2 + C3_2 \ \} + \mathrm{CJ}.$$

b. $lock_{11}$ obtains the lock first.

In this situation, $lock_{21}$ must wait until $lock_{11}$ releases the lock, before it can obtain the lock and $lock_{12}$ must wait for $lock_{21}$ to release the lock, so the time cost is equal to :

$$TC = \max \{C1_1 + CL1_1 + C2_1 + CU1_1 + \max \{C3_1, \ CL1_2 + C2_2 + CU1_2 \ \} + CL2_1 + C4_1 + CU2_1 + C5_1, \ C1_2 + CL1_1 + C2_1 + CU1_1 + CL1_2 + C2_2 + CU1_2 + C3_2 \ \} .$$

From the analysis of previous example, we find that some cases can be reduced to the basic parallel structure 1 in Section 2, while other cases are more complicated due to the conflicts between lock nodes. In the following sections, we focus on the cases that all lock nodes are in conflict, since these cases cannot be reduced to the basic parallel structures in Section 2 and we need to derive a new technique to do the time analysis for these cases.

3.2. Parallel Computation Structure 1

In Fig. 5, we present the first parallel computation structure where only a single path contains two lock nodes. We assume that all of the op_{i1} nodes have the same time costs, any two lock nodes are in conflict, and a lock node is independent of any lock node outside of the parallel structure. For those paths containing one lock node, the time

cost can be derived by using equation (2). For the paths contain two lock nodes, since the second lock node must wait for all of the first lock nodes to release the locks, then it can obtain the lock, the time cost of these paths will contain the time costs of two lock nodes and the waiting time of the second lock. This leads to our first theorem.

Theorem 1: Suppose that we are given the parallel structure as shown in Fig. 5. Assume that all of the op_{i1} nodes have the same time costs and let :

time cost of $op_{i1} = C1_i$;

time cost of $op_{i2} = C2_i$;

time cost of $op_{i3} = C3_i$;

time cost of $lock_{i1} = CL1_i$;

time cost of $unlock_{i1} = CU1_i$;

time cost of $lock_{i2} = CL2_i$;

time cost of $unlock_{i2} = CU2_i$;

time cost of *fork* = *CF*;

time cost of *join* = *CJ*;

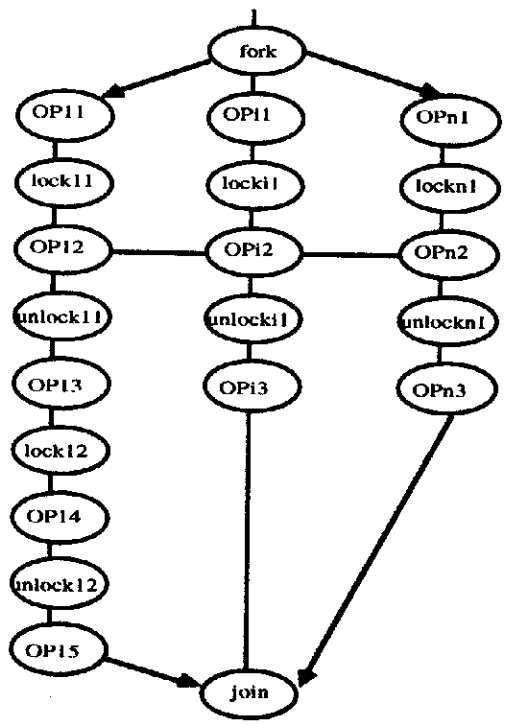

Fig. 5. Parallel computation structure 1

Then, the time cost of this parallel structure is equal to :

$$TC = CF + \frac{1}{n!} * \sum_{i=1}^{n!} \max_{1 \le j \le n} e(i_j) + CJ \qquad (5)$$

where $e(i_j) = \begin{cases} C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + C3_{ij} & \text{if } i_j \neq 1 \\ C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) \\ + max\{C3_1, \sum_{k=j+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} \\ + (CL2_1 + C4_1 + CU2_1) + C5_1 & \text{if } i_j = 1 \end{cases}$

and $(i_1, i_2, \ldots, i_n)$ is the i'th permutation of $(1, 2, \ldots, n)$.

Proof : Suppose that the *n lock* nodes obtain locks in the order $(i_1, i_2, \ldots, i_n)$. There are two possible cases to consider.

* **Case 1:** The lock node $lock_{ij}$ is in the path having only one lock node. In this case, estimating the time cost of this path is the same as that of equation (2). Since all of the op_{i1} nodes have the same time costs and $lock_{i1}$, $lock_{i2}$, $lock_{i3}$, ... , $lock_{ij-1}$ obtain locks before $lock_{ij}$, the time cost of the parallel path containing $lock_{ij}$ is :

$$C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + C3_{ij}$$

* **Case 2:** The first lock node $lock_{ij}$ is in path 1, which has two lock nodes. Since all of the op_{i1} *nodes* have the same time costs and $lock_{i1}$, $lock_{i2}$, $lock_{i3}$, ... , $lock_{ij}$ obtain locks before $lock_{ij}$, the time cost of the path containing $lock_{ij}$ right after it releases the lock and right before reaches op_3 is:

$$C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}).$$

Before path 1 can obtain the second lock, it must wait for all other first locks to be released, so the time cost from op_3 to op_5 is:

$$max\{C3_1, + \sum_{k=j+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + (CL2_1 + C4_1 + CU2_1) + C5_1.$$

Therefore, the time cost of the path containing two lock nodes is :

$$C1_{ij}+\sum_{k=1}^{j}(CL1_{ik}+C2_{ik}+CU1_{ik})+$$
$$max\{C3_{1,}+\sum_{k=j+1}^{n}(CL1_{ik}+C2_{ik}+CU1_{ik})\}+(CL2_1+C4_1+CU2_1)+C5_1.$$

End of proof □

3.3. Parallel Computation Structure 2

In Fig. 6, we present the second parallel computation structure, where the two paths both contain two lock nodes. We assume that all of the op_{i1} nodes have the same time costs, op_{13} and op_{23} have the same time costs, any two lock nodes are in conflict, and a lock node is independent of any lock node outside the parallel structure. For those paths containing one lock node, the time cost can be derived by using equation (2). For a path containing two lock nodes, since the second lock node must wait until all of the first lock nodes release locks, before it can obtain the lock, the time cost of this path will contain the time costs of two lock nodes and the waiting time of the second lock. Since there are two paths containing two lock nodes, we must consider the possible permutations of the second lock nodes. There are two possible order of the second lock nodes, $lock_{12}$ precedes $lock_{22}$ or $lock_{12}$ follows $lock_{22}$. This leads us to the second theorem.

Theorem 2: Suppose that we are given the parallel structure as shown in Fig. 6. Assume that all of the op_{i1} nodes have the time costs and op_{13} andop_{23} nodes have the same time costs.

Then, the time cost of this parallel structure is equal to

$$TC = CF+\frac{1}{n!}*\sum_{k=1}^{n!}{}^{max}_{1\le j\le n}e(i_j)+CJ \qquad (6)$$

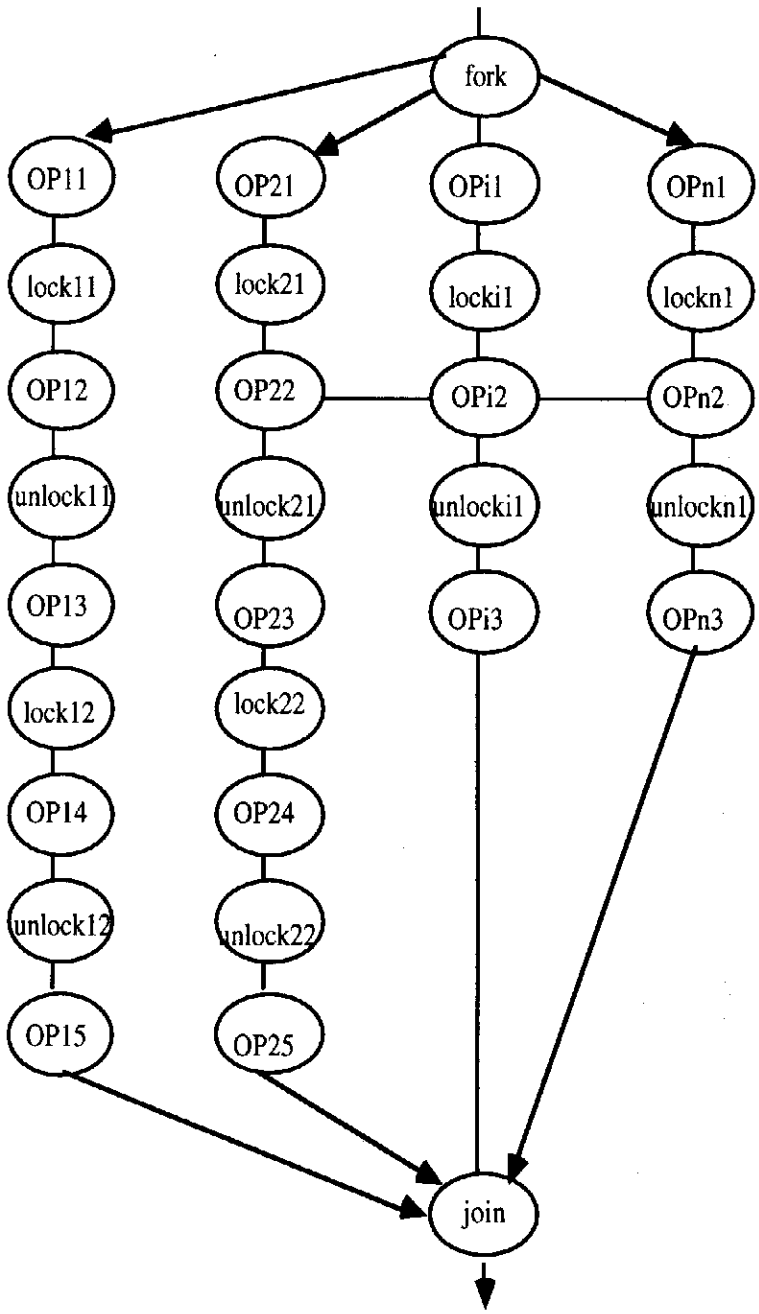

Fig. 6. Parallel computation structure 2

where $e(i_j)$ =

$$
\begin{cases}
C1_{ij} + \sum_{k=1}^{j} (C\,L1_{i\,k} + C2_{i\,k} + C\,U1_{i\,k}) + C3_{ij} & \text{if } i_j \neq 1 \;\&\; i_j \neq 2 \\
\text{Case 1: } lock_{12} \text{ precedes } lock_{22} & \\
\begin{cases}
C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + & \\
max\,\{C3_1, \sum_{k=j+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} & \\
+ (CL2_1 + C4_1 + CU2_1) + C5_1 & \text{if } i_j = 1 \\
C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + & \\
max\,\{C3_1, \sum_{k=j'+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} & \\
+ (C\,L2_1 + C4_1 + CU2_1) + & \\
(CL2_2 + C4_2 + CU2_2) + C5_2 & \text{if } i_j = 2 \;\&\; i_{j'} = 1
\end{cases} & \\
\text{Case 2: } lock_{22} \text{ precedes } lock_{12} & \\
\begin{cases}
C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + & \\
max\,\{C3_2, \sum_{k=j+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} & \\
+ (CL2_2 + C4_2 + CU2_2) + C5_2 & \text{if } i_j = 2 \\
C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + & \\
max\,\{C3_2, \sum_{k=j'+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} & \\
+ (C\,L2_2 + C4_2 + CU2_2) + & \\
(CL2_1 + C4_1 + CU2_1) + C5_1 & \text{if } i_j = 1 \;\&\; i_{j'} = 2
\end{cases} &
\end{cases}
$$

and $(i_1, i_2, ..., i_n)$ is the i'th permutation of (1, 2, ..., n).

Proof: Suppose that the n lock nodes obtain locks in the order $(i_1, i_2, ..., i_n)$. Consider the following two cases.

* **Case 1:** The lock node $lock_{ij}$ is in the path having one lock node. In this case, the way to estimate the time cost of this path is the same as that of equation (2). Since all the op_{i1} nodes have the same time costs and $lock_{i1}$, $lock_{i2}$, $lock_{i3}$,..., $lock_{ij-1}$ obtain locks before $lock_{ij}$, the time cost of the parallel path containging $lock_{ij}$ is:

$$C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + C3_{ij}$$

* **Case 2:** To estimate the time costs of the paths containing two lock nodes, we must consider two possible situations.

1. Assume $lock_{12}$ obtains the lock first, i.w., $lock_{12}$ precedes $lock_{2\,2}$ .Let i_j =1, $i_{j'}$ =2, then examine the first lock node, $lock_{ij}$, in path 1. Since all of the op_{i1} nodes have the same time costs and $lock_{i1}$, $lock_{i2}$, $lock_{i3}$,..., $lock_{ij-1}$ obtain locks before $lock_{ij}$, the time cost of the path containing $lock_{ij}$ right after it releases the lock and right before reaches op_3 is:

$$C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) \cdot$$

Before path 1 can obtain the second lock, it must wait for all other first locks to be released, so the time cost from op_3 to op_5 is :

$$max\{C3_j, \sum_{k=j+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + (CL2_j + C4_j + CU2_j) + C5_j \cdot$$

The time cost of the path containing $lock_{ij}$ node is :

$$C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) +$$

$$max\{C3_j, \sum_{k=j+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + (CL2_j + C4_j + CU2_j) + C5_j$$

Therefore, the time cost of the path containing $lock_{ij}$ node is :

$$C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) +$$

$$max\{C3_{j_1}, \sum_{k=j+1}^{n}(CL1_{i_k}+C2_{i_k}+CU1_{i_k})\}+(CL2_1+C4_1+CU2_1)+(CL2_2+C4_2+CU2_2)+C5_2$$

2. Assume $lock_{22}$ obtains lock first, i.e., $lock_{22}$ precedes $lock_{12}$ with $i_j = 1$ and $i_{j'} = 2$, then consider the first lock node $lock_{ij'}$ in path 2. Using steps similar to the previous situation, we derive the time cost of the path containing $lock_{ij'}$ node as:

$$C1_{i'_j}+\sum_{k=1}^{j'}(CL1_{i_k}+C2_{i_k}+CU1_{i_k})+$$

$$max\{C3_{j_1}, \sum_{k=j+1}^{n}(CL1_{i_k}+C2_{i_k}+CU1_{i_k})\}+(CL2_1+C4_1+CU2_1)+C5_1$$

Therefore, the time cost of the path containing $lock_{ij}$ node is :

$$C1_{i'_j}+\sum_{k=1}^{j'}(CL1_{i_k}+C2_{i_k}+CU1_{i_k})+$$

$$max\{C3_{j_2}, \sum_{k=j'+1}^{n}(CL1_{i_k}+C2_{i_k}+CU1_{i_k})\}+(CL2_2+C4_2+CU2_2)+(CL2_1+C4_1+CU2_1)+C5_1$$

End of proof □

3.4. Parallel Computation Structure 3

In Fig. 7, the thire parallel computation structure where there are m paths that contains two lock nodes is given. We assume that all of the op_{i1} nodes have the same time costs, all of the op_{i3} nodes have the same time costs for i=1 to m, any two lock nodes are in conflict, and a lock node is independent of any lock node outside the parallel structure. For those paths containing one lock node, the time cost can be derived by using equation (2). For a path containing two lock nodes, since the second lock node must wait until all of the first lock nodes have releasesd the lock, before it can obtain

The lock, the time cost of this path will contain the time costs of two lock nodes and the waiting time of the second lock. Based on our assumptions, we must consider all possible orders of the lock nodes. Once the order of the first lock is fixed, the order of the second lock is fixed too. Therefore, we need to consider n ! permutation of the first lock nodes. This leads us to the third theorem.

Theorem 3: Suppose that we are given the parallel structure as shown in Fig. 7. Assume that all of the op_{i1} nodes have the same time costs and all of the op_{i3} nodes have the same time costs for i=1 to m. then, the time cost of this parallel structure is equal to :

$$TC = CF + \frac{1}{n!} * \sum_{i=1}^{n!} \max_{1 \le j \le n} e(i_j) + CJ \tag{7}$$

where $e(i_j) = \begin{cases} C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + C3_{ij} & \text{if } i_j \text{ only contains one lock} \\ g(i_j, a_l) & \text{if } i_j \text{ only contains two lock, } 1 \le l \le m \end{cases}$

and $g(i_j, a_l) = C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + \max \{C3_{al}, \sum_{k=f+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\}$

$$+ \sum_{p=1}^{l} (CL2_{ap} + C4_{ap} + CU2_{ap}) + C5_{al}$$

with $(i_1, i_2, ..., i_n)$ is the i'th permutation of (1,2,...,n) and $(a_1, a_2, ..., a_m)$ is the order of the second locks, and $a_l = i_f$.

Proof: Suppose the n lock nodes obtain locks in the order $(i_1, i_2, ..., i_n)$. Consider the following two cases.

* **Case 1:** The lock node $lock_{ij}$ is in the path having one lock node. Estimating the time cost of this path is the same as that of equation (2). Since all of theop_{i1} nodes have the same time costs and $llock_{i1}$, $lock_{i2}$, $lock_{i3}$, ..., $lock_{ij-1}$ obtain locks before $lock_{ij}$, the time cost of the parallel path containing $lock_{ij}$ is:

$$C1_{ij} + \sum_{k=1}^{j} (CL1_{ik} + C2_{ik} + CU1_{ik}) + C3_{ij}.$$

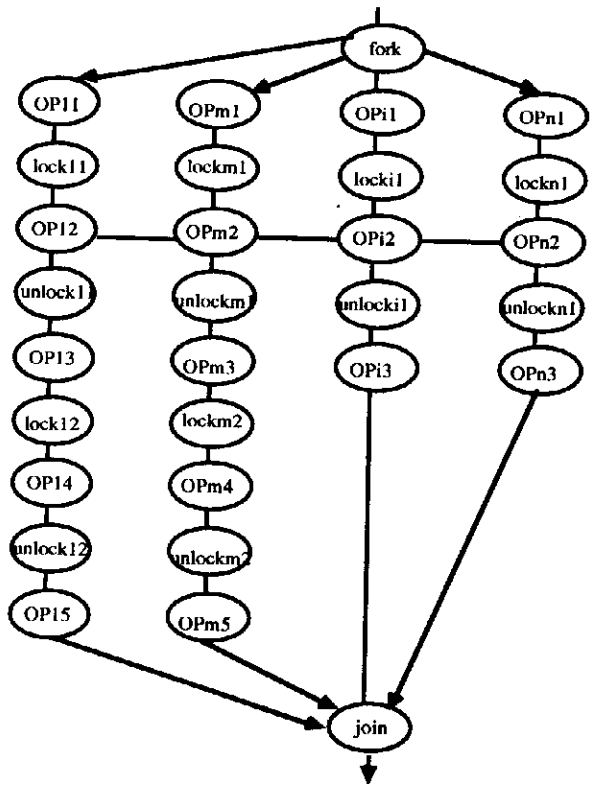

Fig.7. Parallel computation structure 3

* **Case 2:** The paths contain two lock nodes. Assume $(a_1, a_2, ..., a_m)$ is the order of the second locks which is determined by the order of the first locks, i.e. $(i_1, i_2, ..., i_n)$, and $a_1 = i_f$. Let $l = 1$. Consider the path containing the lock node $lock_{ij}$, and $lock_{a1}$. Since all the op_{i1} nodes have the same time costs and $llock_{i1}$, $lock_{i2}$, $lock_{i3}$, ... , $lock_{ij-1}$ obtain locks before $lock_{ij}$, the time cost of the path containing $lock_{ij}$ right after it releases the lock and reaches op_3 is:

$$C1_{ij} + \sum_{k=1}^{f} (CL1_{ik} + C2_{ik} + CU1_{ik}) \cdot$$

Before this path can obtain the second lock, it must wait for all other first locks to be releases, so the time cost from op_3 to op_5 is:

$$\max \{C3_{a1}, + \sum_{k=f+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + (CL2_{a1} + C4_{a1} + CU2_{a1}) +$$

The time cost of the path containing $lack_{a1}$, nodes is:

$$C1_{ij} + \sum_{k=1}^{f} (CL1_{ik} + C2_{ik} + CU1_{ik}) +$$

$$\max \{C3_{a1}, \sum_{k=f+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + (CL2_{a1} + C4_{a1} + CU2_{a1}) + C5_{a1}$$

$$= C1_{ij} + \sum_{k=1}^{f} (CL1_{ik} + C2_{ik} + CU1_{ik}) +$$

$$\max \{C3_{a1}, \sum_{k=f+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + \sum_{p=1}^{1} (CL2_{ap} + C4_{ap} + CU2_{ap}) + C5_{a1}$$

Let $l = m - 1$. Then, the time cost of the parallel path containing $lock_{ij}$ and $lock_{am-1}$ will be:

$$C1_{ij} + \sum_{k=1}^{f} (CL1_{ik} + C2_{ik} + CU1_{ik}) +$$

$$\max \{C3_{a1}, \sum_{k=f+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + \sum_{p=1}^{m-1} (CL2_{ap} + C4_{ap} + CU2_{ap}) + C5_{am-1}$$

Consider $l = m$, then for the parallel containing the $lock_{ij}$ and $lock_{am}$, since all of the op_{i3} nodes have the same time costs and ($lock_{a1}$, $lock_{a2}$, $lock_{a3}$, ..., $lock_{im-1}$ obtani locks before $lock_{am}$, the time cost of the parallel path containing $lock_{am}$ is:

$$C1_{ij} + \sum_{k=1}^{f} (CL1_{ik} + C2_{ik} + CU1_{ik}) +$$

$$\max \{C3_{a1}, \sum_{k=f+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + \sum_{p=1}^{m-1} (CL2_{ap} + C4_{ap} + CU2_{ap}) .$$

$$+ (CL2_{am} + C4_{am} + CU2_{am}) + C5_{am}$$

$$= C1_{ij} + \sum_{k=1}^{f} (CL1_{ik} + C2_{ik} + CU1_{ik}) +$$

$$\max \{C3_{a1}, \sum_{k=f+1}^{n} (CL1_{ik} + C2_{ik} + CU1_{ik})\} + \sum_{p=1}^{m} (CL2_{ap} + C4_{ap} + CU2_{ap}) + C5_{am}.$$

End of proof □

3.5. Parallel Computation Structure 4

In Fig. 7, the fourth parallel computation structure where are m paths that contain two lock nodes is given. We assume that all of the op_{i1} nodes have different time costs, all of the op_{i3} nodes have the same costs for i=1 to m, any two lock nodes are in conflict, and a lock is independent of any lock node outside the parallel structure. For those paths containing one lock node, the time cost can be derived by using equation (3). For a path containing two lock nodes, since the second lock node must wait for all of the first lock nodes to release the lock, before it can obtain the lock, the time cost of this path will contain the time costs of two lock nodes and the waiting time of the second lock. Based on our assumptions, there is only one possible order of the first lock nodes, namely, the second lock nodes obtain the locks in sequential order. This leads us to Theorem 4.

Theorem 4: Suppose that we are given the parallel structure as shown in Fig. 7. Such a parallel structure has all of the properties of the computation as given in Theorem 3, except that the time cost of all op_{i1} nodes are different. Without loss of generality, assume that $C1_1$,< ... < $C1_i$ < ... $C1_n$, and ($C1_1$ + $CL1_1$ + $C2_1$ + $CU1_1$ + $C3_1$) > $C1_n$, i.e. $lock_{11}$ obtains the lock first, then $llock_{21}$, and so on and $lock_{12}$ obtains the lock after $lock_{n1}$ have released the lock.

Then, the time cost of this parallel structure is equal to:

$$TC = CF + \max_{i,j} e(i, j) + CJ \tag{8}$$

$$\text{where} \quad e(i, j) = \left\{ \begin{array}{ll} C1_i + \sum_{k=i}^{j} (CL1_k + C2_k + CU1_k) + C3_j & \text{if } j \text{ contains one lock} \\ C1_i + \sum_{k=i}^{j} (CL1_k + C2_k + CU1_k) + C3_j & \\ + C3_i + \sum_{k=i}^{j} (CL2_k + C4_k + CU2_k) + C5_j & \text{if } j \text{ contains two locks} \end{array} \right\}$$

Proof: Consider the following two cases.

* **Case 1:** The lock node $lock_{ij}$ is in the path having one lock node. Estimating thetime cost of this path is the same as that of equation (2). So the time cost of the path containing one lock node is :

$$max_{1 \le i \le j} \{ C1_i + \sum_{k=i}^{j} (CL1_k + C2_k + CU1_k) + C3_j \}.$$

* *Case 2*: For the path containing two lock nodes, the time cost for first lock is:

$$C1_i + \sum_{k=i}^{j} (CL1_k + C2_k + CU1_k) + C3_j.$$

The time cost for the second lock is:

$$C3_1 + \sum_{k=1}^{j} (CL2_k + C4_k + CU2_k) + C5_j .$$

Overall, time cost of the path containing two lock nodes is:

$$max_{1 \le i \le j} \{ C1_i + \sum_{k=i}^{j} (CL1_k + C2_k + CU1_k) + C3_j$$
$$+ C3_1 + \sum_{k=1}^{j} (CL2_k + C4_k + CU2_k) + C5_j \}$$

End of proof □

4. Example

A example from [10,pp.116-120] is selected to illustrated how the application of the proposed approach to derive the time cost of a parallel computation structure. This example is to find the city closest to Beaverton, Oregon by using Cartesian coordinates, and then printing out the name and distance of the city from Beaverton. One way to solve this problem is to have n processes performing the calculation at the same time, where each process computes the distance from one city to Beaverton. The general structure of the computation is illustrated in Fig. 8 and the computation of the i th process is shown in Fig. 9.

All the operations in the computation are specified as follows.

init:short_dist = 999999999;
cal_dist: calculate the distance between two cities;
lock1: read and write locks on short_dist;
mod_dist: read the short_dist and modify the short_dist;
unlock1: read and write locks on short_dist;
lock2: write lock on city name;
mod_name: modify the city name;
unlock2: write lock on city name;
print: print out the nearest city name and the shortest distance;

Assume the time costs of start and end are zero, and the time costs of all the other nodes are as follows:

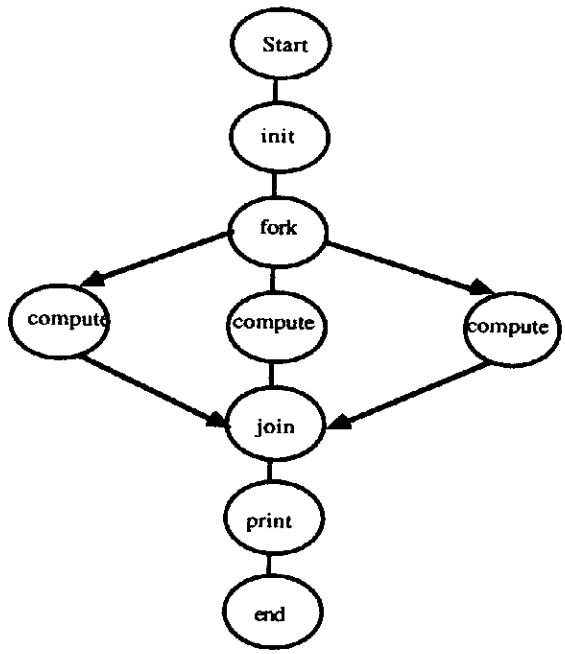

Fig. 8: Computation structure for calculating the nearest city problem

init - - - C_{init}

fork - - - C_{fork}

cal_dist - - - C_{cai}

$$\text{lock1} \text{ - - - } C_{\text{lock1}}$$
$$\text{mod\_dist} \text{ - - - } C_{\text{mod}}$$
$$\text{unlock1} \text{ - - - } C_{\text{unlock1}}$$
$$\text{lock2} \text{ - - - } C_{\text{lock2}}$$
$$\text{mod\_name} \text{ - - - } C_{\text{mod\_name}}$$
$$\text{unlock2} \text{ - - - } C_{\text{unlock2}}$$
$$\text{join} \text{ - - - } C_{\text{join}}$$
$$\text{print} \text{ - - - } C_{\text{print}}$$

Based on the Theorem 3, we can derive the time cost of this example as:

$$TC = C_{init} + C_{fork} + \frac{1}{n!} * \sum_{i=1}^{n!} \max_{1 \le j \le n} e(i_j) + C_{join} + C_{print}$$

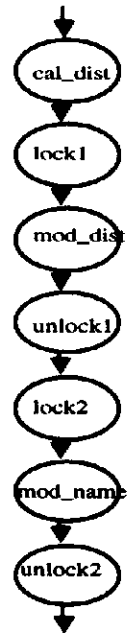

Fig. 9. Computation structure of i'th process

where

$$e(i_j) = g(i_j, a_l) = C_{calij} + \sum_{k=1}^{f} (C_{lock1_{ik}} + C_{mod_{ik}} + C_{unlock1_{ik}}) +$$
$$\sum_{k=f+1}^{n} (C_{lock1_{ik}} + C_{mod_{ik}} + C_{unlock1_{ik}}) + \sum_{p=1}^{l} (C_{lock2_{ap}} + C_{mod\_name_{ap}} + C_{unlock2_{ap}})$$

and $(i_1, i_2, ..., i_n)$ is the i 'th permutation of (1, 2, ..., n) and $(a_1, a_2, ..., a_m)$ is the

order of the second locks, and $a_j = i_f$. In this example, since the second lock node must wait until all of the first lock nodes have been released the lock, before it can obtain the lock, the time cost of this example contains the costs of the two lock nodes and the waiting time of the second lock.

5. Concluding Remarks

For a parallel computation structure, it is difficult to measure the time cost, since communication occurs between operations in a parallel structure. In this paper, we assume that all communications are done via shared memory and our approach is based on the modified computation structure model with communication facilities [1]. A locking technique is used to manipulate the accesses to the shared data. Two new types of nodes, lock and unlock nodes, are added to the model to achieve locking. The time cost to the lock nodes is very difficult to analyze, because of the uncertainties that occur when more than one lock nodes in parallel require conflict locks on a shared item. Previous work has already derived the time costs of a set of special parallel computation structures [2,6]. In this paper, we expanded the investigation towards more general parallel computation structures. In cases of uncertainty we consider all possible oredering (scheduling) methods and evaluate the time cost of each case, then we take the average of these different results. However, it is possible to search for the best scheduling policy that reduces the execution time [10]. It is also possible to follow our approach for cases when the second communication node of a branch can compete with the first communication nodes of other branches.

References

[1] Qin, B. ; Sholl, H. and Ammar, Reda A. "Micro Time Cost Analysis of Parallel Computations." *IEEE Transactions on Computers*, Published by the Institute of Electrical and Electronics Engineers, Inc., Vol. 40, No. 5, (1991), 613-628 .

[2] Qin, B. "Performance Analysis of Parallel Computation." *Ph.D Dissertation, Computer Science & Engg. Dept.*, The University of Connecticut, (1987).

[3] Ammar, R. A. and Qin, B.*"A Technique to Derive Time Costs of Parallel Computations."* COMPSA88, the IEEE Computer Society's Twelfth Internatinal Computer Software & Applications Conference, Chicago, IL., Oct. 1988.

[4] Qin, B. "PPAS: a Performance Analysis Tool for Software Designes, MS thesis." *Technical Report CSE/CARC-TR-84-4, Computer Science & Engineering Department*, the University of Connecticut, (1984).

[5] Tancenbaum, A. S.*"Operating Systems: Design and Implementation."* Eaglewood Cliffs, New Jersey: Prentice-Hall Inc.,(1987).

[6] Ammar, Reda A. ; Ramamurthy Sanjay, and Qin B."*Towards Time Cost Analysis of a General Parallel Structure in Shared Memory Environments.*" Proceedings of the ISMM International Conference on Parallel and Distributed Computing and Systems, Oct. 1990.

[7] Ammar, Reda A. ; Hindam, T. and Darwish N. "*Time Cost Analysis of a Parallel Structure with Multi-communication Nodes in Each Branch.*" The 5th International Conference on Parallel and Distributed Computing and Systems, Pittsburgh, Pennsylvania, October 1, 1992.

[8] Hindam, T. ; Darwish, N. and Ammar, Reda A. "Time Cost of a Parallel Structure with Multi-communication Nodes in Conflict." *Egyptian Computer Journal*, Published by the Institute of Statistical Studies and Research, Giza, Egypt, 1993.

[9] Osterhaug A. (Ed.), *Guide to Parallel Programming on Sequent Computer Systems*, Prentice Hall, 1989.

[10] Mohamed, R. ; Daradshti N. ; Ammar Reda A. and Fergany, T. A." *Optimizing the Time Cost of Parallel Structures by Scheduling Parallel Processes to Access the Critical Sections.*" International Conference on Computing and Information ICCI'92, Toronto, Ontario, Canada, 1992 .

تحليل تكلفة الزمن للتراكيب المتوازية متعددة نقاط الاتصال في بيئة ذاكرة مشتركة

رضا عمار
كلية علوم و هندسة الحاسب ، جامعة كونيكتي كت
كونيكتي كت ٠٦٢٦٩-٣١٥٥ الولايات المتحدة الأمريكية

(قدم للنشر في ١٩٩٣/٠٦/٠٧م، وقبل للنشر في ١٩٩٤/٠٦/٢٠م)

ملخص البحث . في هذه الورقة يتم تحليل تكلفة الزمن لمجموعة من التراكيب الحاسوبية المتوازية . يبنى هذا التحليل على افتراض أن الواحدات البرمجية تتصل مع بعضها البعض عبر ذاكرة مشتركة وأن هناك ميكانيكية حجز تنظم استخدام المتغيرات المشتركة . في عمل سابق قمنا بتطوير طريقة لحساب تكلفة الزمن لمجموعة من التراكيب المتوازية . في هذه الورقة نوسع العمل السابق ونعطي طريقة جديدة تشمل تراكيب حاسوبية متوازية أكثر عمومية .