計算機アーキテクチャ　107－10
（１９９４．　７．　２１）

予約タイムスタンプを用いた分散型データフローマシンに関する一考察

永瀬　宏　　鈴木　洋

金沢工業大学情報工学科

この論文では、分散型データフローマルチプロセッサを提案する。従来の方式はタイムスタンプでタイムワープ機構を用いて大域時間を構成していた。しかし、大域時間を示すのにデータフローマシンでは命令は非同期的に実行され時間の巻き戻しが難しく厳密な大域時間の実現は困難である。従って本論文では送信メッセージに予め受信時に期待するタイムスタンプの予約値を挿入しておくプロセス間通信の手順を提案する。メッセージに予約タイムスタンプとデータフローマシンのタグを利用して大域時間を構成しメッセージが正常に受け渡される方法を検討した。時間の巻き戻しがなく大域時間を構成する方法を、チャネル設定フェーズ、通信フェーズ、解放フェーズを含め紹介していく。

A distributed dataflow machine which uses time stamp

Hiroshi NAGASE　　Hiroshi SUZUKI

Department of Information and computer engineering
Kanazawa Institute of Technology
7-1 ougigaoka nonoichi ishikawa 921 Japan

In this paper, we consider a distributed dataflow multi-processor. In a Neumann processor, realize a global time is realized by time stamp. However, in a dataflow machine, it is difficult to realize a strict global time, since, instructions are asynchronously executed, and it is difficult to achieve roll back. Hence, In this paper, we consider the method of the process communication that insert the reservation value of time stamp, expected in a receive message, in a send message beforhand. These time stamps are easily realized as a part of tags, originally used as colors of operands.

1. Introduction

In a distributed processing system, many processes concurrently operate and thus accomplish high performance. However, among mutually related processes, concurrency control is necessary to assure correct execution of processes. If one supposes existence of a central process that is responsible for time control, many problems can be avoided which are specific to distributed systems. However, in the central controlled system, performance bottleneck often arises, and a failure of center process may causes a serious trouble of a total system. Therefore, in a distributed system, concurrency control had better be decentralized.

For Neumann processors, there exist many researches about concurrency control.[1][2] This paper discusses concurrency control of dataflow machines. Since a dataflow machine is classified to a tag machine, we utilize time stamps. Precisely, 'colors ' are used for distinguishing concurrent processes (or functions), wheras 'time stamps' are used for realizing a global clock. The latter clock is obtained by message handling.

In this paper, we consider dataflow multi-processors. Process communication and mutual exclusion are realised using time stamps. Since in a dataflow machine, instructions are executed asynchronously, it is necessary to correctly identify senders and receivers at process communication. It is also to be noted that asynchronous execution of instructions inhibits winding back of time stamps. Hence, to attain globally ordered time stamps are not easy in a dataflow machine. These problems are considered in the following sections.

2. Existing methods

A method of process communication for a dataflow machine was proposed by Yamada et al[3] which uses a special hardware called a structured memory. A structured memory is composed of a memory and a queue. In this memory, read and write operations happen asynchronously. If a read operation precedes a write operation, a read operation can wait for completion of a write operation in the following way.

(1) Common areas between two processes are reserved in a memory for control of process communication. Receiver process writes, in that area, his color and an entry address of received message. At the same time, a flag, called a presence bit, is put on a memory to signify validity of color and entry address.

(2) Sender process reads a color and an entry address written in a structured memory. Then, a presence bit is put off. Send message is tagged with a color of a receiver process, and is send to a specified entry address.

(3) If a sender tries to read a structured memory in advance that a receiver writes to that memory, the read request waits for completion of a write operation at a structured memory.

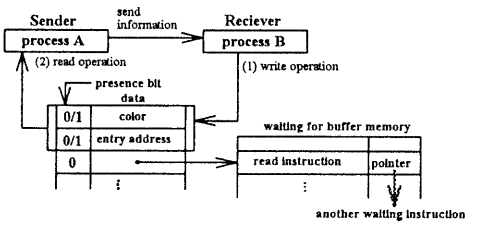

Fig.1. Process communication using a structured memory[3].

For mutual exclusion, a ' test and set ' instruction is used in a Neumann processor. A problem of this method is a memory access neck caused by repeated test instructions. There are other methods that prevent repetition of common memory accesses. However, these methods usually broadcast occupation of a common memory, and this announcement augments the number of process communications.

To overcome these problems, an autonomous method of mutual exclusion was proposed for a dataflow machine.[3] As shown in Fig.2, the method uses control processes and an arbitration token. In an entrance control process, access requests for a common memory are queued. When an arbitration token enters the process, the first request in the queue is taken and is sent to a procedure, in which memory accesses are operated. After the operation is over, an arbitration token is again returned to the

entrance process.

The method, so far stated, is called a ' color suppress matching method,' since an arbitration token can pair with any access requests regardless of their colors. The remarkable feature of this method is that it reduces access load of a common memory. Another point is that a request process can continue the execution of his process after he sent an access request, since an access is autonomously permitted at common memories. However, in this method, one cannot regulate the order of accesses in a specified manner (e.g. order of events related to accesses). Introduction of time stamps, presented in the following section, intends to accomplish sequential access of a common memory.

3. Process communication

3.1 Assumed processor architecture

In the following sections, a multi-processor is supposed, where each processor is a dataflow machine. At execution time, several processes exist in each processor, and every process is composed of several functions. To achieve parallel processing, each function is tagged with a unique color. Here, colors are issued at a tag manager, which is located at every processor. For simplicity, we suppose that the number of colors are infinite.

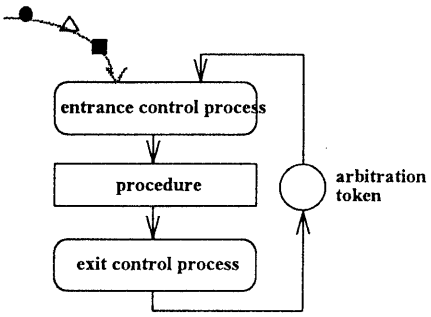

Fig.2. Mutual exclusion.

3.2 Necessity of time stamp

Process communication is defined as message transfer from a sender process to a receiver process. Usually a send request and a receive request are not synchronized. Then, there are three approaches to cope with the difference of timing. The first approach is to use the function of operand waiting inherent to dataflow machines. In this case, message is directly sent to an instruction like an ordinary operand. One difference from usual operand transfer is that a sender and a receiver have different colors. Hence, a sender must send his message tagged with a receiver's color. Since a receiver may do function calls at any time, it is not easy for a sender to know receiver's color. The second approach is a contention method. Process communication shown in section 2 takes this approach, and for this objective, a channel is introduced. Physically, a channel corresponds to a structure memory and a queue. The features of this method are summarized below.

(1) A structure memory plays a role of color transformation. However, colors of a sender and a receiver must be mutually pre-known, and must be fixed until the termination of process communication.

(2) A receiver can preset a receive request. Hence, a receiver address can be determined at the execution time.

(3) A queue is used for waiting a send message when a receive request is delayed. Since messages are accepted by FIFO algorithm, one cannot control the order of messages.

The third approach uses time stamps, and is a basis of this paper. Generally, time stamps offer timing information in a distributed system.(4) The concept of time stamps, here, is similar to those conventional time stamps. However, the proposing time stamps have also a function of relating a pair of send and receive messages. Precisely a send message includes a time stamp that will be used in a return message. Hence a time stamp is a flag which discriminates a return message. For this objective, another solution may be a use of color. However, as shown in the first approach, colors are insufficient since they are dynamically changed at function calls.

3.3 Proposed process communication

An overview of process communication is illustrated at Fig.3. and three steps included here

are explained now.

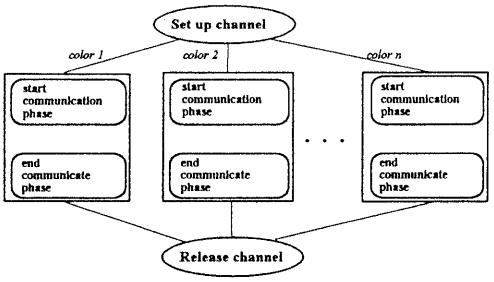

Fig.3. Overview of process communication.

(1) Set-up-channel phase

Initially, a channel must be set up between two processes A and B. Precisely, process A prepares a structured memory shown in Fig.4.

CH-number: N_A			
reserved time stamp	message color	receive color	recieve address

Fig.4. Structure memory of process A.

Now, let us explain each item in Fig.4.

- CH-number ••• means the number NA which identifies process A.
- Reserved time stamp ••• means an expected time stamp which will be used in a return message from process B.
- Message color ••• means an expected color which will be used in a return message from process B.
- Receive color ••• means an internal color of process A, which will be used to receive a message from process B.
- Receive address ••• means an instruction of process A, which will accept a message from process B.

After process A finishes preparing his structure memory, he sends a channel request including a channel number NA to process B. Then, process B similarly prepares a structured memory, and returns a channel number NB. When process A communicates with several processes, he prepares only one structured memory, and sends channel requests to all other processes.

(2) Start-communication phase

Now, process A sends a start-communicate command to process B. For this sake, in advance, process A determines the values of following parameters.

Reserved time stamp t_{RA}••• being calculated as

$$t_{RA} = t_{sA} + 1$$

where t_{SA} is a time stamp of a command to be sent. The value of t_{sA} is determined by a tag manager, which is introduced as a color handling process in the beginning of this section. Precisely,

t_{sA} = (the newest t_{rA} which has been already registered) +1

Receive color C_A being determined simply as a present color C_{pA} of process A. If there is a possibility of a function call, another color must be hunted and reserved.

Receive address ••• being the first entry address of a received message, and must be fixed in advance of process communication.

After these parameters are registered at a structure memory of a channel N_A, process A sends a start-communicate command of Fig.5.

CH-number N_B	command	color C_{pA}	time stamp(t_{sA})

command: start-communicate

Fig.5. Start-communicate command.

Then, process B fulfills a structure memory of channel N_B as follows.

Reserved time stamp t_{RB} ••• being calculated by a tag manager as

$$t_{RB} = t_{sB} + 1$$

where t_{sB} is a time stamp of a command to be sent, and

$$t_{sB} = \max \{ t_{sA}, t_R \} + 1.$$

Here, t_R is a present t_{RB}, the newest reserved time stamp before receiving a message of process A

Manager color ••• being C_{pA}.

Other parts ••• being the same as process A.

After these parameters are registered at a

structure memory of a channel N_B, process B sends an ACK command of Fig.6, with $t_{rA} = t_{sA} + 1$.

CH-number N_A	command	color C_{rB}	time stamp(t_{rA},t_{sB})

command: ACK

Fig.6. ACK command.

(3) communication phase

Process A sends an information message of the format in Fig.7.

CH-number N_B	command	color C_A	time stamp(t_{sB},t_{sA})	Data

command: information frame

Fig.7. Information message.

A structure memory of process A is fulfilled as in the start-communication phase. Process B also sends a message similar to Fig.7.

Example

Let us illustrate an example of a time sequence in Fig.8.

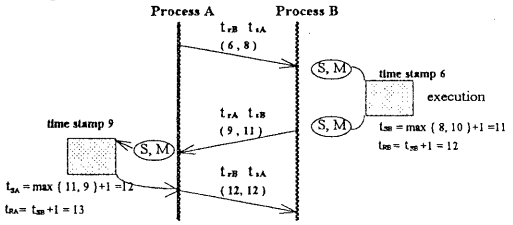

Fig.8. An example of process communication.

Firstly, process A waits for a message reserving a time stamp 6, though the newest reserved time stamp is 10. The situation happens when process B communicates with other processes. After receiving a massage, process B calculates t_{sB} and t_{RB}. The explanation of the remaining part is abbreviated.

4. Discussions

4.1 Internal processing with time stamps

Though an idea of time stamps is well known, features of this paper come from time stamp reservation. By the reservation, a sender can accurately receive a return massage from a receiver. However, in order to accomplish the reservation, each process must transfer a time stamp at execution time of instructions.

Then, there arises a problem that time stamps should be matched, as well as colors, when an instruction of a dataflow program fires. This is not so easy, because there are several cases when time stamps are not fixed deterministically as in Fig.9. In these cases, time stamps must be forcibly changed, and how to change values is dependent on a system designer.

Introduction of time stamps at execution time has also been studied by Nishikawa et al(5), They pointed out that parallel computing of stream data are possible with time stamps.

4.2 Tag manager

A tag manager is assigned for each processor, and handles colors and time stamps for all processes included in a processor. Hence, a tag manager must operate sequentially.

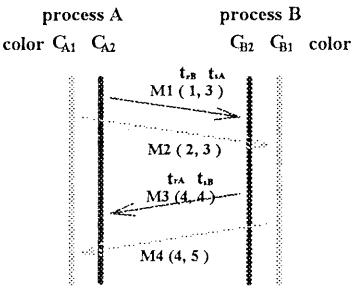

Fig.10. Simultaneous process communications.

To explain the necessity of sequential operations, let us consider a time sequence of Fig.10.In this figure, simultaneously two

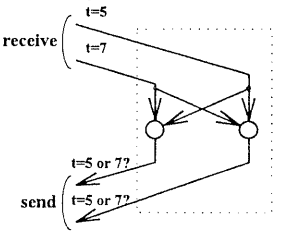

Fig.9. Indeterministic values of time stamps.

communications are proceeded with different colors. In detail, two functions exist at process A, and they share a common channel, since a structure memory is assigned to each process. Hence, process A receives two messages M3 and M4 with the same t_{rA} (=4) from process B.

To distinguish M3 and M4 at process A, two

methods may be applied.

(1) Identify M3 and M4 by colors. For this sake M3 is colored with C_{B2}, and M4 is colored with C_{B1}.
(2) Assigne different values to t_{sA}'s of M1 and M2.

In the first method, start communication phase must run at every function call, which becomes an overhead of process communication. Hence, a color C_S of message had better be fixed to an initial value assigned at start-communication phase. Then, the second method must be applied, and to assign a. unique value to t_{sA}, a tag manager must determine a new time stamp sequentially.

4.3 Effects

With the use of time stamps, the following effects seem to be obtained.

(1) A lost message shown in Fig.11 can be detected.Basically, a timer is set to a reserved time stamp of a structure memory, and time out is detected. For this timer, a global clock is necessary, and time stamps can play a role of a global clock.

(2) Message disordering can be corrected at a receiver process as shown in Fig.12.

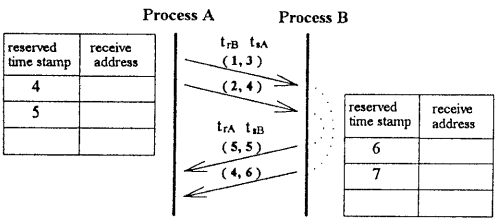

Fig.12. Correction of message order.

Since reserved time stamp t_r is assigned to each message, a destination address of a received message is obtained from a receive address of a structure memory.

(3) Multi process communication is possible as illustrated in Fig.13.

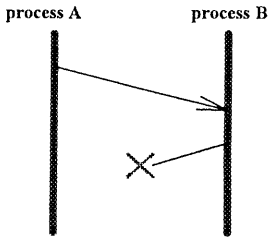

Fig.11. Lost message.

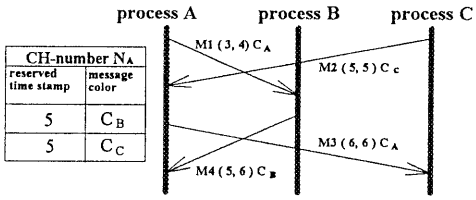

Fig.13. 1:n communication.

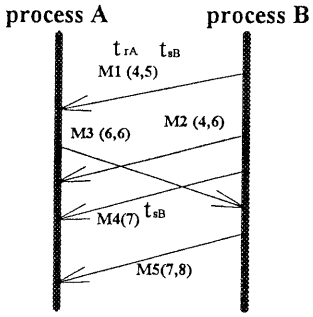

Fig.14. Continuous message sending.

Here, process A communicates with process B and process C concurrently. Moreover, a structure memory of CH-number N_A is shared with process B and process C. Hence, continued messages with reserved time stamp of 5 are not distinguished by a channel nor by a time stamp, but are separated by colors.

(4) Continuous message sending is permitted with some modifications of communication methods. Let us consider a time sequence of Fig.14.

Process B sends two messages M1 and M2 having the same t_{rA} (=4). These are return messages for a preceding message from process A. Moreover, process B sends a new message M4 with t_{sB} being 7. If process A sends a message M3 with t_{sA} being 6, process A must identify which of M4 or M5 being a return message of M3. This distinction can be easily done by checking t_{rA}

included in each message.

(5) Mutual exclusion is already explained in section 2. Using time stamps, enhanced access control is possible.

Fig.15 illustrates an access manager of a common memory.

Difference from Fig.2 is a function of entrance control process. In the new method, read and write access conditions are registered. Here, color is used, if necessary, to limit an accessible process. However, more important constraints are time stamps t_{RE} and t_W, which enable the following access rules.

(1) Read access message with t_{sA} (process A is supposed to access) is aborted if $t_{sA} < t_W$. Else the read access id accepted, and t_{RE} is incremented as

$$t_{RE} = t_{RE} + 1.$$

(2) Write access message with t_{sA} is aborted if $t_{sA} < \max \{ t_{RE}, t_W \}$. Else, the write access is accepted, and t_W is incremented as

$$t_W = t_W + 1.$$

As shown in Fig.8, time stamps t_{sA}, t_{sB}, ••• of messages approximate a global clock when process frequently communicate with each other. Therefore, by using time stamps, one can avoid illegal operations like the case when a delayed read operation is executed on data which are already renewed.

Though time stamps are effective for mutual exclusion, one had better be careful to use them. The first point is that a time stamp is not a strict hardware clock. Colors, included in Fig.15, may simplify the problem, since a real sequential clock is obtained in a single color (= function).

The second point is "what is a basis for an access order?". Let us consider a situation shown in Fig.16.

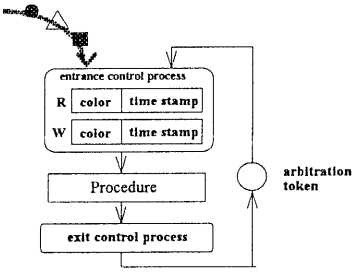

Fig.15. Mutual exclusion with time stamps.

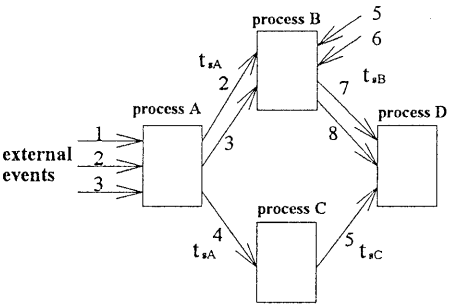

Fig.16. Transfer of time stamps.

Here, external events are input to process A, and messages are transferred to process D. Since messages pass through different processes B and C, the order of time stamps at process D does not coincide with the order of external events. Then, a problem arises that process D should obey the order of global clock t_{sB} and t_{sC}, or the order of initial events. If the latter order is inherent, the order of external events must be included, as third time stamp, in each message. These two points stated above are important but remained as further studies.

5. Conclusion

This paper discusses process communication for a distributed dataflow machine. The proposed method uses time stamps to constitute a logical clock which approximates a global clock when processes frequently communicate with each other. Especially the reservation of a time stamp which will be used in a return message is effective for an accurate message transfer. To show the effects, several discussions are presented including message reordering and mutual exclusion. Finally, as a remained problem, the relations between a time stamp and an order of external events are

considered. Deadlocks and fail-safeness are other important problem which must be studied in future.

Acknowledgment

The authors wish thank Mr.Tawa and other students who discuss with us for this work.

Reference

(1) "Special section on distributed processing", Jour. of Info. Proc. Soci. of Japan, vol. 28, No.4, 1987.

(2) "Special section on algorithms for fault-tolerant distributed systems", Jour. of Info. Proc. Soci. of Japan, Vol.34, No.11, 1993.

(3) Shigeki Yamada, at al, "Design and evaluation of a dataflow-controlled switching system which exploits all forms of parallelism inherent in the switching processing.", Kluwer Academic Publishers, to be appear.

(4) Yonezawa et al,"Method and representation",Iwanami, 1992.

(5) Nishikawa, Terada, Asada, "A Data-Driven Schema Incorporating History Sensitinity.", ,Vol. J66-D No.10, pp.1169-1176,1983.