フォールトトレラントな分散Fleng処理系の実現*

1T－1

尚 軍　青柳龍也　有山正孝†
電気通信大学‡

1 はじめに

1.1 Commited Choice 型言語 Fleng とその分散処理系

われわれは並列プログラミング言語Flengの分散処理系を実装している。FlengはConcurrent Prolog やGHCなどの並列論理型言語とよく似たCommitted Choice 型言語である。Fleng 処理系における各ゴールは、共有変数を通してプロセス間通信を行ない、その具体化待ちによって同期をとる。並列プロセスと考えることも出来るため、並列処理の記述が行ないやすい。処理系の言語はC++を用いた。

1.2 研究目的

疎結合コンピュータで稼働している今のFleng処理系は、他ノードとの通信機構としてデータグラムを使用した為、メッセージの紛失或は、通信路または通信相手ノードの故障が発生すると、全体の処理系がとまってしまう可能性がある。この点に対する改善が今回の研究の目的である。そのアプローチは故障の検出と故障からの復旧の二つに分類される。

- 故障の検出：故障を起こしている部分（相手ノード、通信路、メッセージの紛失）を検出して、アプリケーションに知らせる。
- 故障からの復旧：故障を起こしたノードを正しい状態に復帰させる。ただし、今回の研究は、対象アプリケーションを特定して、効率の良いフォールトトレラントな処理系のの実現を狙う。

2 改善点の概要

2.1 メタコールの実装

故障を検出する為に、以下のメタコールを実装する。

call(**Goal ,Node,Protocol,Control,Result**)

Goal 分配するゴール。

Node 分配先ノード。

Protocol 通信のプロトコル。
プログラマがアプリケーションの種類によって必要なプロトコルを決める。

1. UDP：ノード間の通信がデータグラムに基づいているので、メッセージの到着順序が間違っても影響がない、相対的に高信頼性があまり必要ではない、或は分散実行のノード数が少なくてメッセージ紛失の可能性がとても低いアプリケーションに向いている。
2. UDP++：分散実行のノード数が多数存在する、或は、ノード間物理距離が長い時、メッセージ紛失や順序違いの可能性が高くなる。これに対して、次のように改善する。

   各ノードの送信するメッセージに連続の整数のタイムスタンプを設ける。受信ノードAは送信ノードBから受信したのメッセージのタイムスタンプt_Bを保存している。ノードBから新しいメッセージM_Bが到着した時, もし、
   - タイムスタンプ$t(M_B) = t_B + 1$時、メッセージを処理する。
   - タイムスタンプ$t(M_B) > t_B + 1$時、メッセージM_Bをメッセージテーブルに格納すると同時に、タイマを設定する。タイマが切れるまで、或は、メッセージテーブルが一杯になった場合、タイムスタンプ$t_B + 1$のメッセージが紛失したと判断し、エラーステータスをメタコールの **Result** に反映する。
3. TCP：メッセージの紛失或は順序間違いに対して保証できるから、メッセージの順序関係がとても重要なアプリケーションか、少数ノード間の高信頼性が必要なアプリケーションに向いている。

   制御的に複雑なのは、コネクションの確立と切断のタイミングである。今回とっている方法は、ゴールを新しいノードに分配する時、資源の制限の為、コネクションを確立できない場合は、すでにコネクションを確立しているノードの中で、輸入表と輸出表に参照が存在しなくなっているノードがあれば、そのノードとのコネクションをクローズする。輸入表 (ImportTable) とは、他のノードからの遠隔参照に関する情報を持つテーブルで、輸出表 (ExportTable) は、他のノードへの遠隔参照に関する情報を持つテーブルである。

   これによって、あるノードが故障が発生して、通信ができない時、、その旨をメタコールの **Result** に反映する。

Control コントロールストリーム。**Result** に対して、プログラマによって制御する。(resume,abort,continue) から構成する。

---

*Fault-Tolerant Distributed Implementent of Concurrent Programming Language Fleng

†Jun SHANG, Tatsuya AOYAGI, Masataka ARIYAMA
‡The University of Electro-Communications

Result ゴールの実行結果 (1. success 2. deadlock 3. node_failure 4. message_missing 5. other)。

2.2 対象特定のアプリケーションの故障状態からの復旧.

メタコールの実装によって、メッセージの紛失やノードの故障をプログラマに知らせる機能を処理系に持たせた。この機能を用いて、プログラマがメタコールの Result に基づいて、適当なコーディングを行なって、アプリケーションの故障状態からの復旧を図る。

今回選ぶアプリケーションは、他ノードに分配したゴールが結果を返す以外には他の副作用が全くなく、つまり、複数実行されても、悪影響にはならないアプリケーションである。

N ビットの加法器を考えると、$X = (X_1, X_2, \ldots, X_n)$、$Y = (Y_1, Y_2, \ldots, Y_n)$、N 個のノードによって、各ビットの加法ゴールを並列に生成し、各ノードが自身の計算を終了してから、下位ビットの加算ノードからの入力情報を待つ。入力情報が来たら、自身の計算結果と加算して、上位ビットの加算ノードに情報を送って、終了する。

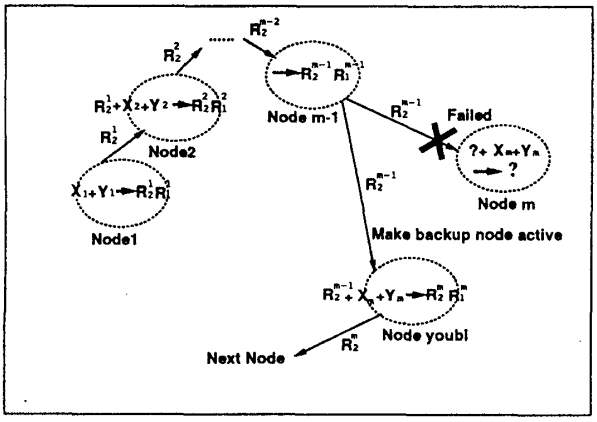

図 1: N ビット加法器分散実行

もし、m 個目ノードが故障が発生したら、m 個目ノードを予備用ノードに切替えて、再実行するだけで、m 個目ノード前まで、ロールバックして、再実行する必要がない仕組みができる。ただし、故障発生したノードが持つセルが他ノードから参照されている場合は、どのようにその参照を再実行したノードに移すかは今後の課題である。

以下に N ビット加法器のプログラムリストの一部を示す。

```
n_add(X1,X2,...,Xn,Y1,Y2,...Yn,
    Node1,Node2,...,Noden,BackupNodeList):-

        do(add_2(X1,Y1,R12,R11),Node1,
                                BackupNodeList),
        do(add_3(RC12,X2,Y2,R22,R21),Node2,
                                BackupNodeList),
        ...,
        do(add_3(RC(n-1)2,Xn,Yn,Rn2,Rn1),Noden,
                                BackupNodeList).
```

```
do(Goal,Node,BackupNodeList):-
        call(Goal,Node,tcp,Cntrl,Result),
        check_result(Result,Goal,BackupNodeList).

check_result(fail,_,_).
check_result(deadlock,_,_).
check_result(node_failure,Goal,BackupNodeList):-
        ask_user(Answer),
        do_answer(Answer,Goal,BackupNodeList).
do_answer(resume,Goal,BackupNodeList):-
        select(Node,BackupNodeList,BackupNodeList1),
        do(Goal,Node,BackupNodeList1).
do_answer(abort,Goal,BackupNodeList).
......
```

3 おわりに

今回の研究ではメッセージの紛失やノードの故障の検出を行なって、アプリケーションに知らせて、更に、Fleng のコーディング上の簡単な対応によって、対象が特定のアプリケーションの故障からの復旧によって、フォールトトレランスの分散 Fleng 処理系を実現する方法を考えた。今後、このシステムを実装し、メッセージ数、通信負荷に対する影響、分散ノード数との関係、効率性などについて評価を行なう予定である。

しかし、今回の対象アプリケーションでは、まだ汎用性が欠けている。復旧の方法に関して、より効率良く、プログラマに対して、より透過的な自動故障復旧のアルゴリズムが必要である。

参考文献

[1] HENRI E. BAl ,JENNIFER G. STEINER ,ANDREW S. TANENBAUM. "Programming Languages for Distributed Computing Systems." *ACM Computing Surveys*,Vol.21,No. 3 ,September 1989

[2] "特集 フォールトトレラント分散システム向けアルゴリズム." 情報処理,Vol. 34 , No. 11, pp.1335-1374 (Nov.199

[3] EHUD SHAPIRO "The Family of Concurrent Logiv Programming Languages." *ACM Computing Surveys*,Vol.21,
No. 3 ,September 1989

[4] 藤原克則 , 青柳龍也, 有山正孝,
"大域的ゴミ集め機能を持った分散ヒープ."
情報処理学会第４７回全国大会論文集,1993