ベクトル時間を用いた分散プログラム用デバッガの実装について*

6 J − 1 2

木下 秀人，木俣 貴博，太田 剛，酒井 三四郎†
静岡大学‡

1 研究背景

同期問題による再現性の欠如は，分散プログラムのデバッグを著しく困難にする．

そこで本研究では，ベクトル時間 [1][2] によって制御される 3 種類の breakpoint を用いた分散プログラム用デバッグシステムについて説明する．

本システムは，半順序関係を持つ事象の履歴を記録，管理し，実行を制御することによりプログラムの振る舞いを再現する．

2 分散プログラム用デバッガの概要

分散プログラムにおいてサイクリカルなデバッグを実現するために，図 1に示すようなシステムを考える．

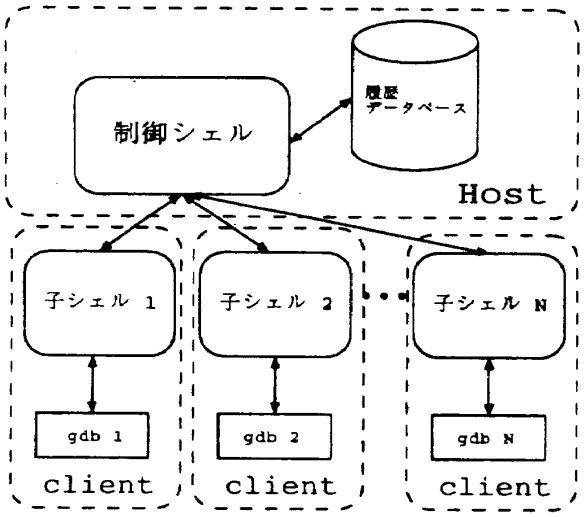

図 1: 本システムの概要

図 1の各要素は以下に示す機能を持つ．

- 履歴データベース

  ベクトル時間による履歴情報を管理し，その履歴から各プロセス毎のメッセージの到着順序を管理する．

- 制御シェル

  - 各子シェルで起動しているプロセスと履歴データベースとの，プロセス毎の履歴情報の対応を管理する．
  - 各子シェルに対して run,continue 等の gdb のコマンドを送る．
  - 子シェルで起動している gdb に入力されたコマンドを一度制御シェルで受け取り，そのコマンドを判断し，全体の流れに影響しないコマンド (print,ptype 等) ならば実行する．
  - 履歴データベースと再実行時のメッセージの着信順序を比較して着信順が異なっているならば，履歴情報に則した順序にメッセージを受け取るように制御する．

- 子シェル

  gdb を実行し，制御シェルからのコマンドを gdb に渡す．また gdb からのコマンドを制御シェルに渡す．

3 制御方法

分散プログラムのデバッグ時には，ユーザからの以下のような要求が想定される．

- ユーザが指定したポイントで，個々のプロセスの状態を確認したい．
- ユーザが指定したプロセスのポイントより後ろで，他のプロセスが予期しない動作をして，ユーザが指定したプロセスの振舞を妨げていないかを確認したい．
- ユーザが指定したプロセスのポイントより前で，誤った振舞をしていないかを確認したい．

これらの要求を満たすために，それぞれ以下に示すような制御方法を定義する．

· break point

ユーザは個々のプロセスに対して自由に break point を置くことができる．

そしてシステムは各プロセスがその breakpoint で停止するように制御する．

---

*Implementation of distributed debugging system based on vector clock

†Hideto Kinoshita, Takahiro Kimata, Tsuyoshi Ohta and Sanshiro Sakai

‡Shizuoka University

・ **possible point**

ユーザは任意の一つのプロセスに possible point を置くことができる．

そしてシステムはそのプロセスを possible point で停止させ，他のプロセスはそのプロセスの停止に影響を受ける所まで，可能な限り実行を継続するように制御する．

・ **minimum point**

ユーザは任意の一つのプロセスに minimum point を置くことができる．

そしてシステムはそのプロセスを minimum point で停止させ，他のプロセスはその minimum point が実行され得るために，最低限必要なだけ実行させるように制御する．

4　実現方法

このシステムを用いて，前述の制御方法をそれぞれ次のように実現する．

・ **break point**

ユーザが指定したポイントまで受信事象の順序を，履歴情報と照合しながら実行を制御する．

・ **possible point**

ユーザによって指定されたプロセスの possible point と他のプロセスの各事象のベクトル時間を比較して，各プロセスで possible point よりも後で発生する事象のうち，最も古い事象を検索する．

そしてそのポイントに breakpoint を設定し，全プロセスでそれぞれ受信事象の順序を，履歴情報と照合しながら実行を制御する．

これを図2を用いて説明する．ユーザがプロセスBのPに possible point を設定すると，ベクトル時間 (1,3,1) よりも後で発生する事象のうち，最も古い事象のベクトル時間を検索する．

この例ではプロセスAではHの (4,5,1)，プロセスCではIの (2,4,3) に breakpoint が設定される．

・ **minimum point**

ユーザによって指定されたプロセスの minimum point と他のプロセスの各事象のベクトル時間を比較して，各プロセスで minimum point よりも先に発生する事象のうち，最も新しい事象を検索する．

そしてそのポイントに breakpoint を設定し，全プロセスでそれぞれ受信事象の順序を，履歴情報と照合しながら実行を制御する．

これを図2を用いて説明する．ユーザがプロセスBのPに minimum point を設定すると，ベクトル時間 (1,3,1) よりも先に発生する事象のうち最も新しい事象のベクトル時間を検索する．

この例ではプロセスAではKの (1,0,0)，プロセスCではJの (0,0,1) に breakpoint が設定される．

これらは，因果先行 [3] より明らかである．

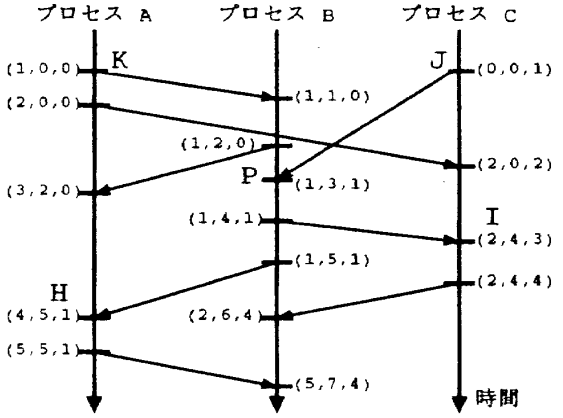

図 2: 分散プロセス間でのメッセージパッシングの様子

5　今後の方針

本稿では，事象履歴による再現性を用いた分散プログラムのサイクリカルなデバッグのためのシステムを定義し，またその際，解析するために必要となる制御方法について定義し，実現手段を示した．

今後の方針として，分散プロセスの実行の流れを効率よく制御する方法について考察し，実装する．

なお本研究の一部は，文部省科学研究費 (課題番号 10780181) の援助のもとに行われた．

参考文献

[1] 白鳥則郎, 滝沢誠: “分散処理”, 丸善 (1996)

[2] Mattern,F.: “Virtual Time and Global States of Distributed Systems”, in Parallel and Distributed Algorithms (Cosnard,M. and Quinton,P.eds.), North-Holland, Amsterdam, pp.215-216(1989).

[3] Lamport,L: “Time,Clocks,and the Ordering of Events in a Distributed System”, ACM,Vol.21,No.7,pp.558-565(July 1978).