7J－3 ユーザが拡張可能なイベント監視機構を持つ分散デバッガの設計と実現

伊藤 隆幸　千葉 滋　猪原 茂和　益田 隆司
東京大学 大学院 理学系研究科 情報科学専攻

1 はじめに

今日ますます重要となりつつある分散アプリケーションは、従来の逐次型アプリケーションと比べ、並列性・非決定性・伝達遅延などの要因により、デバッグの困難が著しく増していることがたびたび指摘されている[3]。それゆえ、効率的なデバッグのためには、分散アプリケーションの性質に特化したデバッガの支援が望まれ、国内外を問わず多数の研究が行なわれてきているが、未だ一般的に広まるには至っていない。

本稿では、当研究室にて現在研究開発中のデバッグ支援システムについて述べる。本システムは、メッセージ交換により協調動作する分散アプリケーションを対象とし、メッセージ送受信などのイベントを監視してデバッグ作業を支援するものであるが、内蔵の小型インタプリタ言語によりデバッガの動作を記述することができ、柔軟性および拡張性を高めている。このようなアイディアはCMUのBEE[1]などにも見られるが、本システムでは、汎用の言語を搭載することにより、イベント監視条件のみならずデバッガのさまざまな挙動を統一的に記述できるシステムを目指している。

2 分散環境でのバグに関する考察

分散アプリケーションにおいては、複数のプロセス(ないしスレッド)がメッセージで通信しながら並列に動作している。個々のプロセスの動作は既存のデバッガで追うことも可能であるが、プロセスの数が増大すると極めて繁雑な作業となる。加えて、プロセス間のメッセージ送受信の様子をグローバルに把握する一般的な手段がないため、デバッグ作業を行なうことが困難となっている。

さて、分散アプリケーションのバグの原因について考えてみると、大別して２つのパターンがあるものと考えられる。すなわち、メッセージやプロセスが予想外の挙動を示すことによるバグと、実行時の微妙なタイミングのずれなどに起因するバグである。本研究では特に前者に着目し、分散アプリケーション開発の初期において頻発するこの種のバグのデバッグを取り扱う。具体的なバグの例をあげると、「送信された(はずの)メッセージが受信されていない」「メッセージの内容や順序が予定と異なる」「同じメッセージが重複している」などがある。こういったバグを効率良く発見することが本システムの目的である。

Design and Implementation of a Distributed Debugger System with Event Monitoring Mechanisms Extensible by the User
ITO Takayuki, CHIBA Shigeru, INOHARA Shigekazu, and MASUDA Takashi
The Department of Information Science, Graduate School of Science, the University of Tokyo

3 デバッガの設計

デバッグのためにプログラムの挙動をトレースする場合、実行の単位、すなわち論理時間の概念が要求される。これは、逐次型のプログラミングモデルにおいては、実行中のプログラムの行やプログラムカウンタの値により自明に与えられていた。しかし、並列性をもった分散プログラムにおいては、これらは適切な尺度とはなり得ない。そこで、本研究では、Lamportの分散イベントの順序付けのモデル[2]に基づき、メッセージ送受信をイベントとして規定される論理時間を考える。すなわち、デバッガは、アプリケーションによるメッセージ通信イベントを単位として実行を制御するものとする。

さらに、デバッガがこれらのイベントに介入し、実行の流れやメッセージを制御することにより、次のような分散デバッグ支援機能を実現することが可能となる。

トレース、ロギング — プロセス間のメッセージ通信の様子を追跡・記録する。

ブレークポイント、ステップ実行 — 各プロセスで発生するイベント(群)が指定の条件を満たしたとき、プロセスの実行を一時停止する。また、イベント単位でプロセスをステップ実行する。

メッセージに対する操作 — 送受信されるメッセージやその内容に対して種々の操作を行い、そのメッセージを受信した時のプロセスの挙動を調べる。

4 実装の方針

本システムは、ユーザにより操作されるデバッガと、アプリケーションのイベントを監視するデバッグ用ラン

タイムライブラリから成り、これらの協調動作によりデバッグ機構を実現する。アプリケーションにリンクされたライブラリは、通信イベントを検出し、デバッガに通知する。デバッガはこれに基づいてユーザに情報を提供するとともに、ユーザからの指令をライブラリに伝達し、アプリケーションの動作を制御する。

ここで、アプリケーションへのデバッガの介入をいかにして実現するかが問題となる。デバッガを利用する場合とそうでない場合とでアプリケーションのソースコードに煩雑な変更が必要となるようでは実用上問題である。しかし、その手間を避けるあまり、特定の機種のOSやライブラリに深く依存するような実装法では、使い勝手は良くても、分散環境における異機種間の移植性・相互運用性を損なうおそれがある。すなわち、アプリケーションの変更は再コンパイル・リンク程度の手間で済み、かつ機種等に依存せずに実現可能な標準的手段によることが望ましい。

これらの点を考慮し、本システムでは次のような実装法を採用した。アプリケーションはC言語で記述されているものとし、デバッグ時にはそのソースコードの先頭部でデバッグ用ヘッダファイルを読み込む。この中のマクロによって、以後のメッセージ通信関連システムコールの呼び出しは、すべて本システムのライブラリの呼び出しに変更される。ライブラリは、本来のシステムコールを呼び出すとともに、イベントを監視・制御する。以上の方法により前述の条件を満たすことができるが、その反面、再コンパイルが不可能な場合(例えば、ソースコードが提供されないツールキットライブラリ等を利用して通信する場合)に関しては適用できない。

5 内蔵言語による記述と制御

デバッグ作業においては、バグの発生箇所・原因を特定することが主眼であるが、この作業にあたって注目すべき対象、すなわちデバッガが監視すべき対象(イベント、メッセージ、状態など)とその内容は、個々のアプリケーションやバグの状況によってさまざまである。また、デバッガによる不必要なオーバーヘッドを抑えるためにトレースの強度を調節したり、実行結果を記録したログを適宜取捨選択して必要な情報のみを提示するといった機能も望まれる。

本システムでは、このような多様な要求に対処する枠組みとして、デバッグシステム中に小型のインタプリタ言語処理系を内蔵し、これによってユーザがデバッガの挙動を記述できるようにし、柔軟性を持たせている。

この言語は、Forthに類似した文法形式およびセマンティクスを持つが、実行時オーバーヘッドを軽減するため、一般的なForthシステムと比べていくつかの点で大きく異なっている。主な相違点は、あらかじめプログラムテキストを中間コードにコンパイルして実行すること、システム定義の基本ワードは再定義不可能な予約語として扱われること、ユーザ定義のワード・変数・配列などはコンパイル時に静的に辞書およびヒープに登録されること、などである。

この言語を用いた記述例を以下に示す。これは、マルチキャスト(16ノード)送信後の受信メッセージを調べ、マルチキャスト対象外のノードからのリプライを検出したらブレークポイントを起動する例である。

```
16 array dest-list
: predicate-on-recv
    msg-buf msg-id-ofs + @
    reply-msg-id = if-then
        0 dest-list 16
        msg-buf dest-node-ofs + @
        lookup not
        if-then 1 break-flag ! endif
        drop
    endif ;
```

6 おわりに

本稿では、メッセージ交換による分散アプリケーションを対象とするデバッグシステムについて、その設計と実装法の要諦を述べた。

現在、SUNおよびHPのワークステーション上で、ソケットによるプロセス間通信を行うアプリケーションのためのプロトタイプシステムが稼働している。

今後は、上記の内蔵言語を用いて記述・実現すべき各種機能について、さらに深く考察していきたいと考えている。ユーザインタフェースの改良、リプレイやグローバルステートのサポート、逐次型デバッガとの連係などが検討課題である。

参考文献

[1] Bernd Bruegge. A Portable Platform for Distributed Event Environments. In *Proceedings of the ACM/ONR Workshop on Parallel and Distributed Debugging*, pp. 184–193, 1991.

[2] Leslie Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. In *Communications of the ACM*, pp. 558–565, 1978.

[3] 関田, 市吉, 西岡, 吉光. 超並列プログラムでのデバギングの考察. 情報処理学会第48回全国大会講演論文集, 第5巻, pp. 133–134, 1994.