5H-7

データ再演と巻き戻し実行を組み合わせた 並列プログラムデバッガのための可視化システム

黒坂　竜之助†　　丸山　真佐夫†
木更津工業高等専門学校†

１．はじめに

並列処理は高い計算能力を実現するために用いられている．コンピュータで並列処理を行うプログラムは並列プログラムと呼ばれ，一般的な逐次プログラムとは区別される．

並列プログラムを開発する際に注意しなければいけない点として，並列プログラムのデバッグが逐次プログラムのそれと比較して難しい点が挙げられる．

並列プログラムのデバッグには，処理内容を把握することを補助し，エラーの発見を助けるツールが必要である．また，そのツールはデバッガ等の他のツールと連携できることが望ましい．

並列プログラムの処理内容を把握するためのツールとしては jumpshot[1)]が挙げられる。しかし、jumpshot はプロセス間の通信が多くなると表示が複雑になり、実用的でないという欠点がある。

そこで本研究では，実用的な並列プログラムの可視化を行い，デバッグを効率的に行うための可視化プログラムの開発を行った．この可視化プログラムは MPI[2)]によって並列動作を行うプログラムを対象とする．また，本研究ではデータ再演法と巻き戻し実行の組み合わせを用いたデバッギングシステムを利用する．可視化対象となるのは主なデータは，データ再演のために保存された通信ログである．

２．デバッグが困難な理由

並列プログラムのデバッグが，プロセッサ単体による逐次プログラムのそれと比較して困難である理由は以下の通りである．

（１） 非決定性

並列プログラムは，同一の条件でプログラムを実行した場合でも，その振る舞いが異なる場合がある．このようなプログラムを非決定的で あるという。

非決定性によって，以前の実行では発生したバグが発生しなくなる，あるいは逆に以前は発生しなかったバグが発生する事が考えられる．このような現象は，デバッグを難しくする要因となる．

A Study of Visualization System for Debugging of Parallel Programs Based on Data-replay and Reverse Execution
†Ryunosuke KUROSAKA, Masao MARUYAMA
Kisarazu National Collage of Technology

（２） 処理の流れの複雑さ

並列プログラムは複数の処理の流れで構成される．それぞれの処理は他の処理と通信を行うため，バグの本来の原因とその発現箇所が異なり、デバッグを難しくする可能性がある。

（３） 計算資源の占有

プログラムの実行に複数のコンピュータを利用する場合、デバッグのためにそれらのコンピュータを長時間占有することは望ましくない。

また、再実行でバグを発現させるまでに時間がかかる場合には、その時間がデバッグの効率を低下させる原因となる。

３．デバッグ手法

2 章で示したように，並列プログラムのデバッグは逐次プログラムのそれと比べて困難であるが，その対処方法はいくつかある．それらの方法のうち，本研究で用いるものについて以下に述べる．

３．１ 再演法

並列プログラムのデバッグ手法として再演法が提案されている．再演法では，プロセス間の通信ログをとり，次の実行からはそのログの内容にしたがって通信を再現する．

本研究では、再演法の一種であるデータ再演法[3)]を用いている。

３．２ チェックポインティングと巻き戻し実行

プログラムの実行のある時点の状態を保存することで，プログラムを巻き戻して(あるいは早送りして)実行することができる．

これを利用してプログラムの任意の箇所を繰り返し実行できれば，デバッグの効率を向上させることができる．

３．３ 可視化

可視化あるいは視覚化は，デバッグを効率的に行うために，プログラムの動作の流れなどを図示することである．

並列プログラムにおいては，プロセス間の通信をグラフ等に表示することで，プロセス間の関係を把握しやすくなる．

本件研究で開発したのはこの可視化のためのプログラムである．

４．可視化システムの概要

本研究で作成した可視化システム(以下，本シス

テム)の概要を以下に示す.

４．１ 可視化の目的

可視化の目的はデバッグの効率向上である．本システムにおいては，並列プログラムの理解補助と，デバッギングシステムとの連携によって効率向上を実現する.

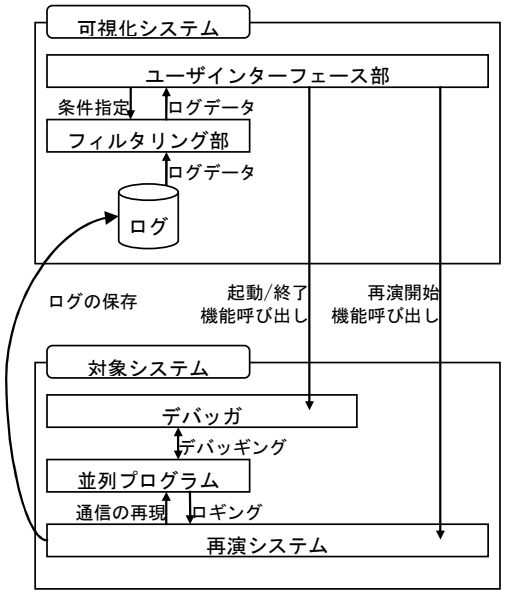

図４．１ システム構成

４．２ システム構成

本システムの全体は，ユーザインターフェース部（以下 UI 部）とフィルタリング部の二つのサブシステムから構成される．システム構成を図示したものを図４．１に示す.

UI 部は，フィルタリング部から渡されるログデータを適切な形式でユーザに示す．また，ユーザからの入力に対応してデバッガの起動や再演実行の開始等を行う.

フィルタリング部は保存されているログを読み込み，入力された条件に従ってフィルタリングを行う．フィルタリング部の実装については，既存のシステムを利用している.

５．可視化プログラムの機能

本システムの主な機能を以下に示す.

５．１ グラフ表示

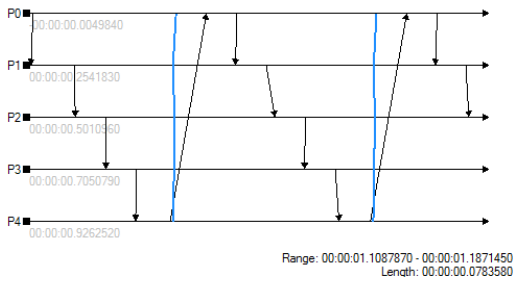

図５．１ グラフ表示の例

ユーザがプログラムを直感的に理解するための手助けとして，グラフ表示を行うことができる．グラフ表示の例を図５．１に示す.

この例では，P_0 から P_4 までの五つのプロセスがメッセージをリレーしている様子が表示されている．また，P_4 から P_0 へメッセージを通信する際にバリア同期を行い，メッセージが一巡したことを全プロセスが確認している.

５．２ フィルタリング条件の指定

グラフ表示の際にはフィルタリングを施すことができる．グラフに表示される情報量はプログラムの規模や実行時間，実行プロセス数によって大きく増加するため，必要なデータだけを表示するためにフィルタリングは必須である.

フィルタリング条件の設定は，UI 部を用いて簡易に行う方法と，テキストによって詳細に行う方法が利用できる．UI 部を用いたフィルタリング条件指定の例を図５．２に示す.

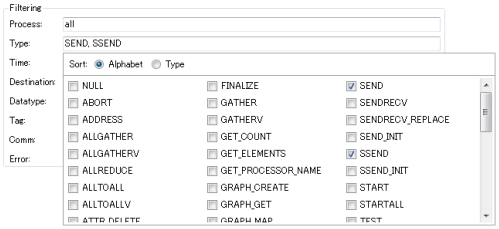

図５．２ フィルタリング条件の指定

５．３ デバッガ・再演システムとの連携

チェックポイントをグラフに表示し，ワンクリックで巻き戻し実行を開始できれば，デバッグの効率が向上できると考えられる.

この機能を実装するためには，デバッガおよび再演システムと通信を行う必要がある．本システムでは SSH 通信によってデバッグ対象のシステムにログインし，コマンドを発行することでこの機能を実現している.

６．おわりに

並列プログラムは計算性能の向上に欠かせない要素であるが，開発にあたってはそのデバッグが大きな問題となる．本研究ではその対策として視覚化とデータ再演・巻き戻し実行を組み合わせた可視化システムを開発した.

今後は実際に様々な並列プログラムのデバッグを行い，より効率的なデバッグができるよう，システムを改良していく必要がある.

参考文献

1) Performance Visualization for Parallel Programs, http://www.mcs.anl.gov/research/projects/perfvis/software/viewers/index.htm

2) Message Passing Interface (MPI) Forum Home Page, http://www.mpi-forum.org/

3) 丸山真佐夫・津邑公暁・中島浩，“データ再演法による並列プログラムデバッギング”，情報処理学会論文誌：コンピューティングシステム，Vol.46,No.SIG12(ACS11),pp.214-225(2005)