# 分散プログラム開発支援ツールの開発

4U－4

黒田澤希†　上山善嗣‡　新井利明†
†日立製作所システム開発研究所　‡日立製作所ソフトウェア開発本部

## 1　背景

分散システムは、ネットワークで接続された複数のマシンが各々機能分担することにより、拡張性やスケーラビリティに優れたシステムを構築できるため、近年重要性が高まりつつある.

分散システム上で動作するプログラムの開発 (分散プログラム開発) は、まず分散したマシンの上で動くクライアントプログラムとサーバプログラムを別々に開発し、それらを関連づけた環境で統合テスト／デバッグを実施するという手順が一般的である. しかし、クライアントプログラムとサーバプログラムが異なるマシンで動作するため、一連の処理の確認およびエラー発生時のデバッグが困難である、という問題がある.

この問題を解決するため、クライアントプログラムとサーバプログラムの動作を関連づけて監視、表示する分散プログラム開発支援ツール StopMotion を提案し、DCE[1][1] 上にインプリメントしたので報告する.

## 2　StopMotion のねらい

分散プログラム開発では、クライアントプログラムとサーバプログラムが異なるマシンで動作するため、従来の単体マシン上でのテスト／デバッグに比べて、以下の問題が生ずる.

**(1)　プログラムの動作把握が困難**　分散プログラムのテストでは、クライアントプログラムからの要求に応じて、サーバプログラムが意図した処理を実行していることを確認する必要がある. しかし、この場合、どのクライアントからの要求に応じてサーバプログラムが実行しているかを把握する手段がないため、テスト作業は多くの工数を必要とする.

**(2)　エラー範囲の特定が困難**　プログラムのエラーが発生した場合には、エラーに関係するプログラムの範囲を特定し、エラーの原因を探る必要がある. 単体マシン上でのデバッグではエラー発生時に動作していたプロセスのスタックの内容から、呼び出したプログラムをたどることでエラー原因となったプログラムが特定できる.

しかし分散システムでは、サーバプログラムでエラーが発生した場合においてもその原因となったクライアントプログラムは別のマシンで動作しているため、上記の方法では原因となったプログラムを特定することは不可能である.

上記のような問題があるため、従来の分散プログラム開発では、個々のマシン上でデバッガを稼働させながら細かい単位で処理を進めてテスト／デバッグする必要があり、多くの工数が必要であった. これらの問題点を解決することが StopMotion のねらいである.

## 3　分散プログラム開発支援ツール StopMotion の概要

### 3.1　StopMotion の機能

StopMotion は、クライアントプログラムとサーバプログラムの動作を関連づけて表示することにより、分散プログラムのテスト／デバッグを容易とするツールである. 主な機能は以下の通り.

**(1)　クライアントプログラムとサーバプログラムの結合表示機能**

クライアントプログラムの動作とそれに関連したサーバプログラムの動作を連結して、一連の処理として表示する. 関連づけは自動的に行われるため、ユーザは対応関係を指定する必要はない.

**(2)　表示の詳細度設定機能**　エラー原因を特定する場合には、徐々に、エラーの発生した範囲を狭めていく方法が一般的である. この目的で、取得する情報量や情報の種類を必要に応じて設定可能としている.

**(3)　エラーイベントの表示機能**　エラーが発生するとき、エラー情報を持つイベントを発行する. このイベントを強調表示する事によって、エラーに関わるスレッドが確認でき、エラーの範囲を特定できる.

**(4)　スレッド表示機能**　マルチスレッド環境においても、各スレッドの動作を区別して表示する. また、並行して動く他のスレッドの発生したイベントの履歴を同時に表示することで、スレッド間の関連を示す.

### 3.2　StopMotion の構成

上記機能を実現するための StopMotion の構成を図1に示す.　StopMotion は、2つのライブラリと2つのプログラムからなる.

#### 3.2.1　イベント発生 DCE ライブラリ

StopMotion では、RPC やスレッドなど、DCE サービスの処理 (イベント) を実行するとき、イベント情報を取得する. イベント情報を取得するために DCE のライブラリに約 300(表1) のフックを入れている. そのた

---

"StopMotion: a Software Development Kit for DCE"
Takaki KURODA, Yoshitsugu UEYAMA, Toshiaki ARAI
Hitachi Ltd.

[1]DCE: Distributed Computing Environment

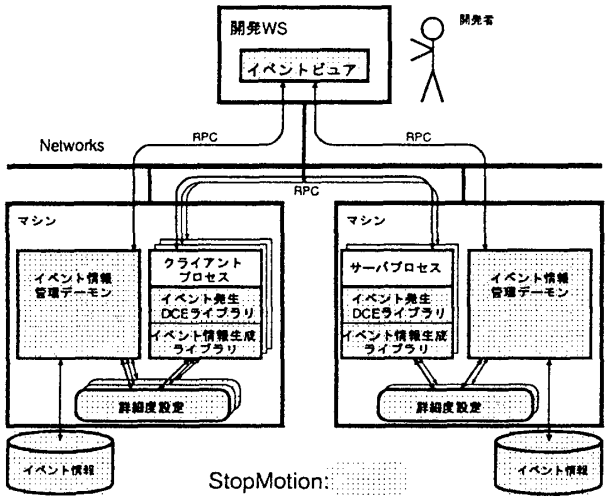


図 1: StopMotion の構成

めユーザプログラムの変更無しにイベント情報を取得できる.

表 1: 主なイベント

| イベント発生場所 | イベント情報内容 |
|---|---|
| 全てのイベントに共通のデータ | スレッド ID, 時刻, イベント ID など |
| サーバの登録 | ポート番号, インタフェース ID など |
| RPC 送受信 | 送信元マシン ID, ポート番号, 受信先マシン ID, ポート番号, コール ID, シーケンス番号, RPC 関数番号, 引数 など |
| スレッド生成 | 新スレッド ID, 初期ルーチンアドレス, 引数 など |
| スレッドの同期／排他制御 | mutex 変数、condition 変数 など |
| シグナルハンドラ | シグナル番号, シグナル発生アドレス, レジスタ内容 など |
| 例外処理 | 例外番号, 例外取得アドレス など |

### 3.2.2 イベント情報生成ライブラリ

フックが呼ばれる時、イベント情報生成ライブラリは各々のイベントに応じたイベント情報を発生する. イベント取得は、イベント取得の詳細度の設定にしたがって、イベント情報の取得を止めたり、詳細な情報を取得できる. 詳細度は、プログラムの起動時に環境変数として指定するか、実行中のプログラムを指定してイベントビュアを使って、プログラム毎に指定を変更することができる.

### 3.2.3 イベント情報管理デーモン

イベント情報管理デーモンは、イベント情報生成ライブラリで取得したイベント情報をファイルに記録したり、イベントビュアからの要求にしたがってイベント情報を転送する.

### 3.2.4 イベントビュア

イベントビュアは開発者の要求にしたがって、各マシンのイベント情報管理デーモンからイベント情報を収集し、ポート番号によってクライアントプロセスとサーバプロセスを関連付けて、各々のプログラムのスレッドの稼働状況を結合して表示 (図 2) する.

図 3に機能と効果をまとめる.

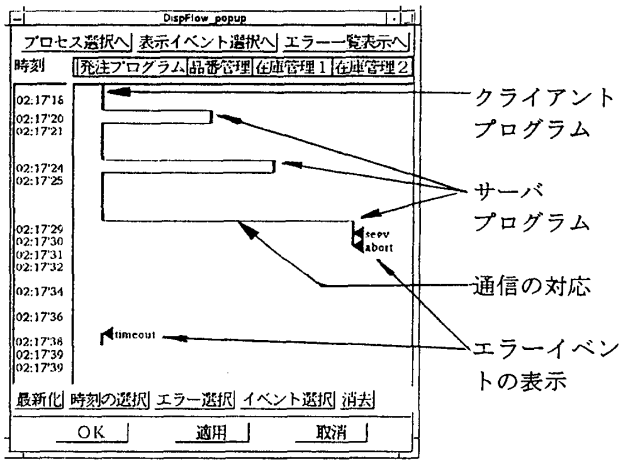


図 2: イベントビュアの画面

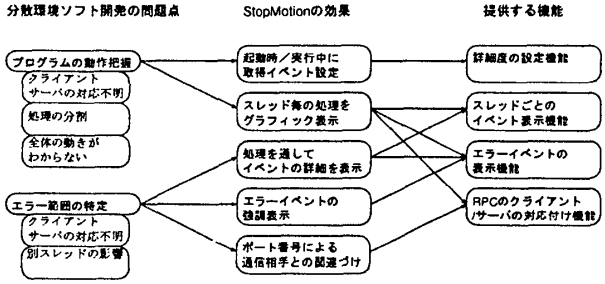


図 3: 機能と効果

## 3.3 StopMotion の利用方法

- **プログラムの動きを知る:** プログラムの処理の遷移をスレッド単位で表示することで、複数のスレッドの関係がわかる.
- **エラー原因の究明:** エラーイベントを表示することで、エラーに関わる処理がわかる.
- **処理性能測定:** イベント情報は時刻を含む. 処理の開始時刻と終了時刻から処理性能を求めることができる.
- **テストのカバレージ:** イベント情報を見ることでテスト済みのパターンを知ることができる.

## 3.4 まとめ

StopMotion はクライアントプログラムとサーバプログラムの対応をとり、また並行に動作するスレッドの関係を表示することにより、分散プログラム開発を支援する.

## 参考文献

[1] "OSF/DCE 入門", 1993, (株) トッパン.