2006－EVA－19 (5)
2006／11／13

Javaプログラム向けメソッドフロー解析ツールの開発

†関洋子、†長須賀弘文、‡吉瀬宏、‡鈴木友峰
†（株）日立製作所 システム開発研究所、 ‡（株）日立製作所 ソフトウェア事業部

Javaプログラムベースでサーバシステムを構築するのが一般的になってきた。また、サーバシステムの性能や信頼性は、ソフトウェアの性能や品質に依存するところが大きい。

そこで、このようなシステム向けに、ひとつひとつのリクエスト処理（トランザクション）に対するソフトウェアの動作手順や、個々の手続きが消費したCPU資源量や処理経過時間の解析を可能にしたJavaプログラム向けメソッドフロー解析機能「JProT」を開発した。JProTを用いることで、ソフトウェア性能のボトルネック抽出や、障害の発生箇所の特定、および障害要因の究明の容易化などが期待できる。本稿では、JProTの機能概要、ならびに効果について論ずる。

Development of Method Flow Analysis Tool for Java Programs

†Yoko Seki, †Hirofumi Nagasuka, ‡Hiroshi Kise, ‡Tomomi Suzuki
†Hitachi, Ltd., Systems Development Laboratory, ‡Hitachi, Ltd., Software Division

It is common to build server systems based on Java programming now. Performance and reliability of server systems depend largely on performance or quality of software.

For those systems, we developed "JProT", a method flow analysis function for Java programs, which enables analysis of software behavior responding to individual request processing (transaction), and CPU usage or elapsed time consumed by individual procedure.

We expect that JProT is useful to locate bottleneck in software performance, detect software failures, and discover the causes of failures.In this report, we introduce features and effects of JProT.

1．はじめに

数多くのサーバシステムが、Javaプログラムによって構築されるようになった。

システム開発では、サービスを開始する前に動作検証テストを実施するのが一般的である。そこでは、開発したプログラムが設計どおりに動作し、性能上の問題がないことや、プログラムに不良が潜在していないことを確認する。こうした検証作業の効率化を図るためには、ソフトウェア動作の詳細を解析できる検証ツールが必要となる。また、こうした検証作業も、単に、サーバ上でのソフトウェアの動作順序を解析するといった形態ばかりではなく、個々の業務サービスに対応させたソフトウェアの動作順序を解析できる検証ツールがあると、より一層の効率化を図ることができる。

現在、Javaアプリケーションプログラムの分野においても、多数のプロファイリングツールが存在し[1]~[3]、性能測定や動作解析など数多くの場面で活用されている。これらのツールでは、サ

ーバ内部でのアプリケーション内部で実行される各メソッドの平均実行時間などといった性能上の指標を容易に掌握でき、性能上の阻害要因となり得る部分の究明に役立てることができる。

しかし、多種多様なリクエストに対応できるようなシステムを構築し、運用させるためには、既存の機能を用いた解析だけでは対応しきれないケースが発生しているのが実態であると考える。

例えば、特定のメソッドに着目した解析においては、既存機能では、モニタリング中に動作したメソッド毎の平均実行時間や平均CPU時間といった統計情報を採取できるが、リクエストの種類に対応したメソッドの動作状況の解析までは困難であった。

また、サーバシステムでは、複数のリクエスト処理（トランザクション）を並行して動作させるのが一般的である。そのため、動作検証の際には、個々のリクエストに対応させて、メソッドの実行順序を解析する必要がある。しかし、既存機能では、モニタリング中に動作したメソッドの実行順序を追跡することができたが、リクエスト処理に対応付けるといったところまでは至っていなかった。

今回、我々は、このような課題を解決すべく、個々のトランザクションの実行毎にその内部挙動を解析し視覚化するメソッドフロー解析機能「JProT」を開発し、機能の有効性を検証することとした。本稿では、JProTの機能概要と効果について論ずる。

JProTの開発にあたっては、OSS(Open Source Software)のJavaアプリケーションサーバを提供しているJBossコミュニティで開発されているJBoss Profiler[1]をベースにした。現在、JProTは、JBoss Profilerの機能の一部として取り込まれており、OSSとしてJBossのサイトから自由に入手・利用可能となっている。

2．メソッドフロー解析機能の開発

2．1　JBoss Profiler

JBoss Profilerは、Javaプログラムの動作ログをファイルに書き込む機能と、その動作ログからソフトウェア動作を解析する機能から構成されるオフライン型のプロファイラである（図　1参照）。JBoss Profilerは、JVMPI(Java Virtual Machine Profiler Interface)により得られた情報をファイルに書き込み、その情報を用いて、メソッド間の呼び出し関係や、各メソッドの延べ処理経過時間ならびに延べCPU時間を解析し表示する。なお、JVMPIにより得られる情報は、オブジェクトやメソッドのイベント情報である。そこに、リクエストごとの識別を取り込んだのがJProTである。JProTは、動作ログからメソッドの開始イベントと終了イベントに関する情報を抽出し対応付けることで、個々のリクエストのメソッドフローを再構築し、視覚化している。

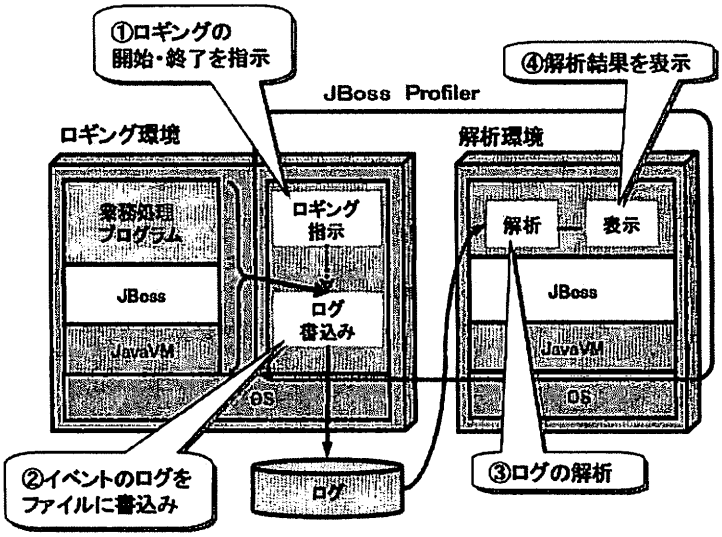

図 1　JBoss Profiler の構成

2．2　JProT の機能概要

JProT は、フロー表示機能と、検索・絞込み機能、ならびに障害解析機能からなる。
以下、それぞれの機能概要を示す。

（1）フロー表示機能

あるトランザクションの内部で実行されるメソッドの実行順序、呼び出し関係などの流れ（フロー）を視覚化する。これにより、同じメソッドで開始するようなトランザクションであっても、実行毎の内部挙動の違いを明らかにできると期待する。

フロー表示機能は、一覧表示画面と詳細表示画面の 2 つの画面を備え、各々、表 1 に示す情報を表示する。

表 1　フロー表示機能の画面

画　面	説　明
一覧画面	同じメソッドで開始するトランザクションの一覧を表示する。 • トランザクションを開始するメソッドの開始時刻 • トランザクションを開始するメソッドの終了時刻 • トランザクション全体の経過時間 • トランザクション内部で実行されるメソッドの個数
詳細画面	一覧画面で選択したトランザクションの内部で実行されるメソッドのフローを表示する。 • メソッドの名称 • メソッドの開始時刻 • メソッドの終了時刻 • メソッドの経過時間 • メソッドの CPU 時間

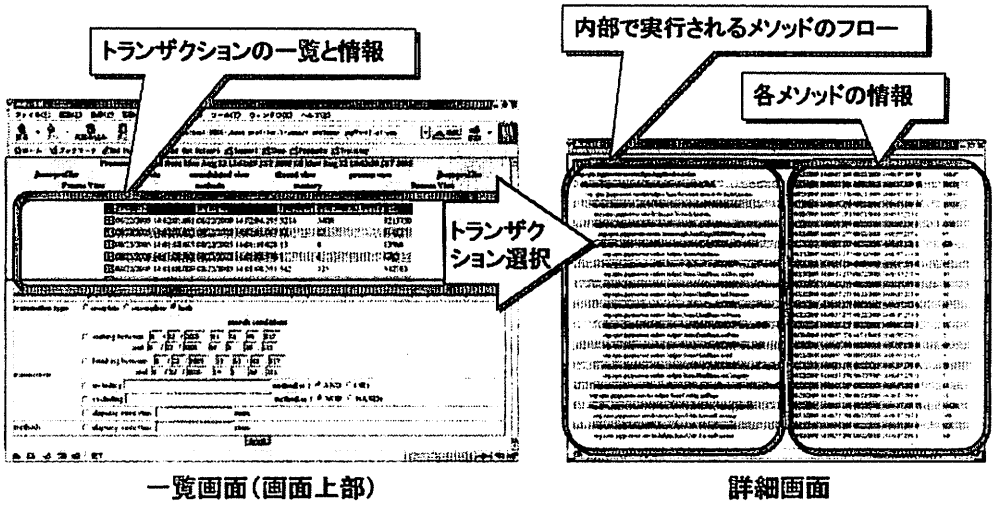

図 2　フロー表示機能

(2) 検索・絞込み機能

多数のリクエスト処理の中から、特定の条件を満たすリクエスト処理を抽出する機能である。この機能により、解析対象のシステムにおいて、キーとなるリクエスト処理を抽出し、ソフトウェアの動作状況を検証する作業が容易になる。

本機能では、抽出条件を設定する画面を用意し、特定時間帯に動作してリクエスト処理や、特定のメソッドが動作するリクエスト処理、システム内滞留時間が長いリクエスト処理などといった条件を設定できる。

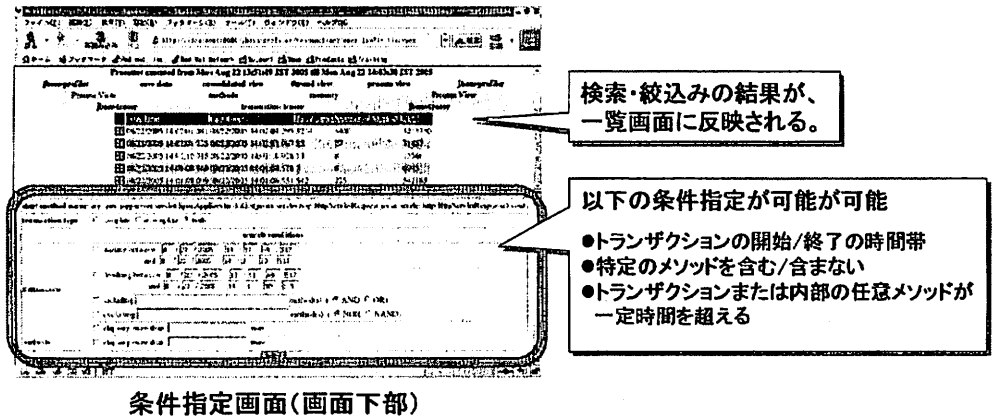

図 3　検索・絞込み機能

（3）障害解析機能

モニタリング中に発生した障害を報告する機能である。

例外が発生した時に動作していたプログラムの特定、ならびに処理内容の明確化は、システム動作中に発生した例外発生要因の究明に大きな手助けになる。さらに、リクエスト処理に対応づけられた処理フローの中に、こうした障害情報を組み込むと、障害要因の究明ばかりでなく、その障害要因の影響範囲の予測にも大きく役立てることができる。

3．JProT を用いた解析

以下、今回開発した JProT を用いた解析例を説明する。

（1）フロー表示機能

解析対象としたシステムは、Java アプリケーションサーバのベンチマークツールである SPECjAppServer2004[4]のアプリケーションである。図 4は、ユーザ処理の 1 つをトランザクションとして解析した結果の一例であり、メソッドのフロー表示の一部を抜粋したものである。

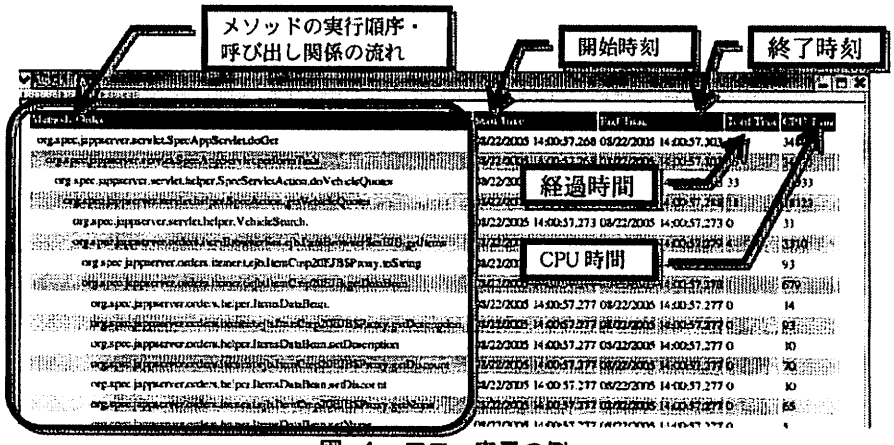

図 4　フロー表示の例

このように、アプリケーション内部の挙動を視覚化でき、統合テストのようなフェーズでの動作検証ツールとして有効であると考える。

（2）検索・絞込み機能

解析対象としたシステムは、ユーザからのアクセス毎に 1 を加算した数値を返す簡易なカウンタであり、与える引数により人為的にソフトウェア障害(RuntimeException)を発生させることが可能なアプリケーションである。ここでは、5 回アクセスし、Servlet の doGet メソッドをトランザクションとして解析する際に、ソフトウェア障害を発生させたトランザクションを絞り込む例を説明する。

図 5は、トランザクション一覧画面に、5 つのトランザクションが表示されている。

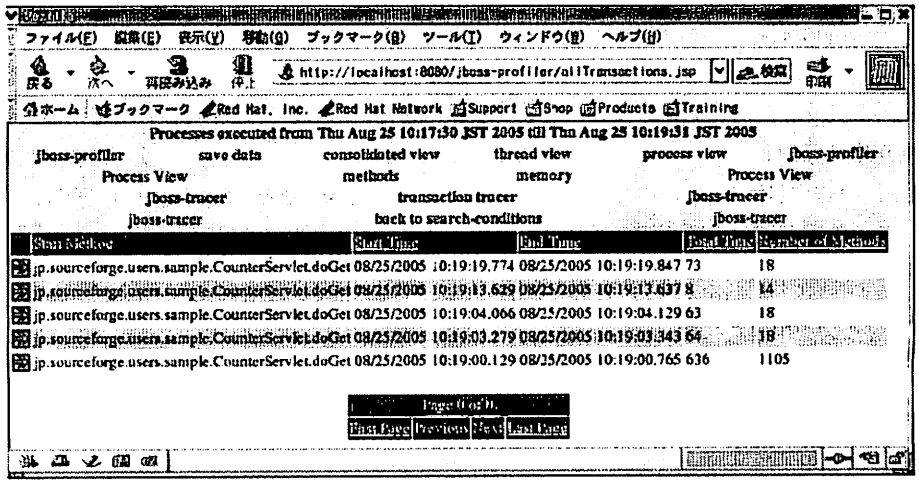

図 5　トランザクションの一覧

得られたトランザクションをベースに、障害を発生させるメソッドの名称と発生する RuntimeException をキーとして絞込みのための条件設定を行い（図 6画面下部）、検索を実行すると、図 6画面上図に示すように、2つのトランザクションに絞り込むことができる。

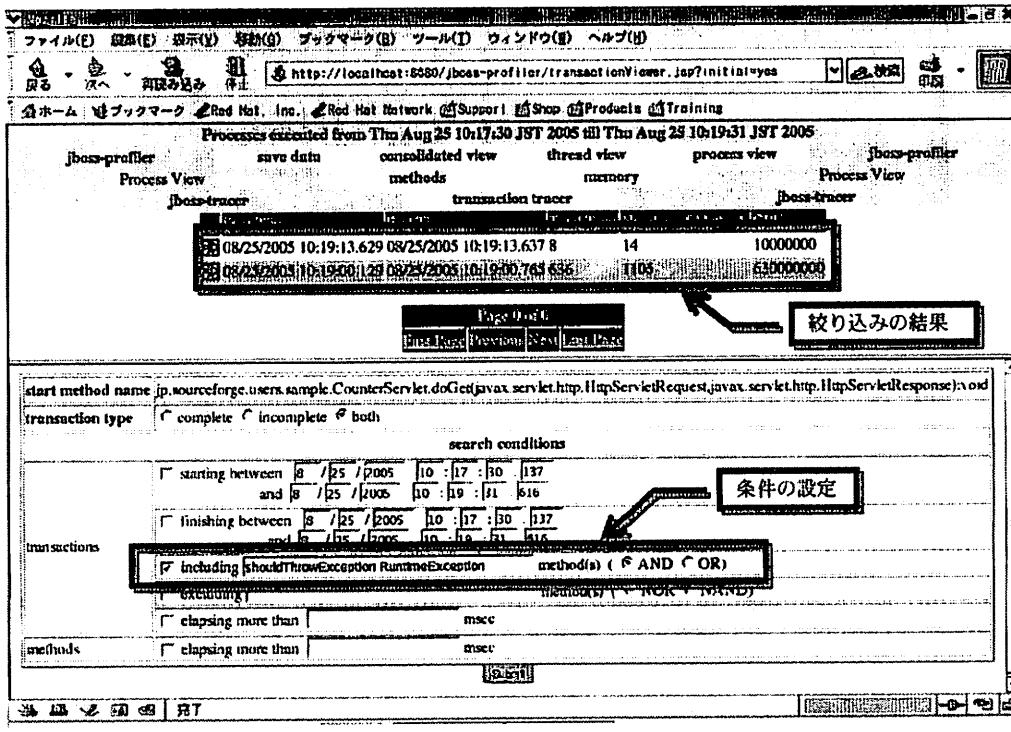

図 6　検索・絞込みの例

このように、検索・絞込み機能を活用することにより、目的とするトランザクションの発見を容易化できる。

（3）障害解析機能

図 7は、（2）で得られた2つのトランザクションから1つを選択し、解析した結果である。

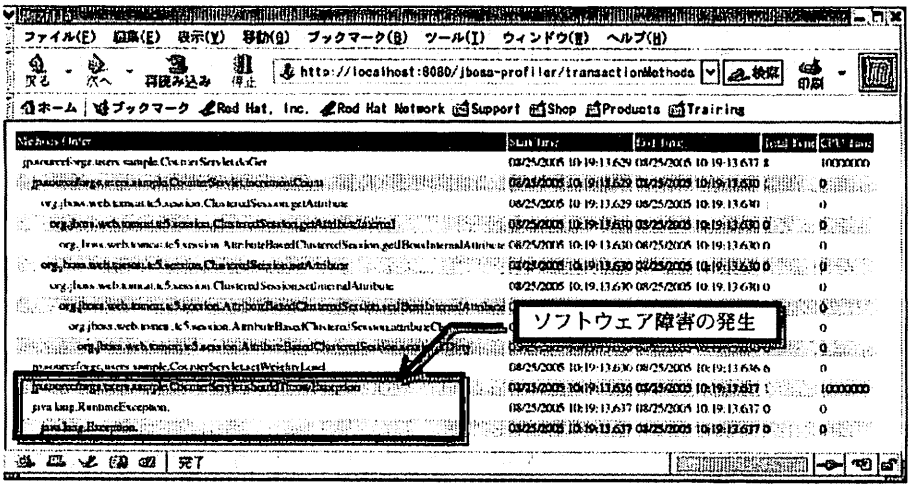

図 7　ソフトウェア障害情報取得の例

このように、JBoss の挙動を視覚化できるとともに、発生した例外をフロー内部に表示でき、JBoss 上のアプリケーション障害発生箇所の発見が容易になる。そのため、どのようなケースにおいて例外が発生しているかを容易に特定でき、障害解析ツールとして有効であると考える。

4．おわりに

一つ一つのリクエストに対応したソフトウェア動作の解析を支援するメソッドフロー解析機能「JProT」を開発した。

JProT は、フロー表示機能と、検索・絞込み機能、ならびに障害解析機能からなる。システム開発時の動作検証などの際に、これらの機能を用いることにより、ソフトウェア性能のボトルネック抽出や、障害の発生箇所の特定、および障害要因の究明の容易化などを図ることができる。これにより、高品質、高性能なシステムの開発に向けた作業効率の向上を図ることができるのではないかと考える。

5．追記

今回の研究開発は、2005 年度上期の独立行政法人情報処理推進機構(IPA)の「オープンソース・ソフトウェア活用基盤整備事業」に関わる委託業務の一環で推進したものである[5][6]。

6．参考文献

[1] JBoss Profiler, http://labs.jboss.com/portal/jbossprofiler

[2] Eclipse profiler plugin, http://sourceforge.net/projects/eclipsecolorer

[3] YaJP, http://yajp.sourceforge.net/

[4] SPECjAppServer2004, http://www.spec.org/jAppServer2004/

[5] 日本 OSS 推進フォーラム　開発基盤ワーキンググループ、
http://www.ipa.go.jp/software/open/forum/development/index.html

[6] 2005 年度「OSS 性能・信頼性評価／障害解析ツール開発」-「Java アプリケーション層の評価」報告書、
http://www.ipa.go.jp/software/open/forum/development/download/051115/web.pdf