

Java の診断を IBM スタイルで: 第 1 回: IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer の紹介

限りない数のダンプ・ファイルを切り抜けて問題の核心をつかむ

[Helen Beeken](#)

Software Engineer
IBM

2013年 8月 22日
(初版 2007年 10月 02日)

[Daniel Julin](#)

Software Engineer
IBM

[Julie Stalley](#)

Software Engineer
IBM

[Martin Trotter](#)

Software Engineer
IBM

Java アプリケーションがますます複雑になった今、Java アプリケーションの問題を診断するのは容易なことではありません。場合によっては外部サービス組織との広範な作業も必要になってきます。そんななか、診断を正しい方向に導く指針があれば時間と経費の両面での節約になります。IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer は、フォーマット済みシステム・ダンプで基本的な分析を行い、簡潔なレポートを生成して次取るべき措置を示すツールです。

この記事では、IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer を紹介し、このツールで診断可能な問題のタイプについての基本情報を提供します。また、IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer を構築するベースとなっているアーキテクチャーについて説明するとともに、このツールの将来の方向性についての考えを述べます。

2013年 8月 22日 — 記事で取り上げるツールの名称を、現在の名称である「IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer」に変更するとともに、「[参考文献](#)」に新しい項目「Java Technology Community」を追加しました。

[このシリーズの他の記事を見る](#)

このトピックに関するスキルを磨いてください

このコンテンツは、皆さんのスキルを漸進的に磨いていくための Knowledge Path の一部です。次のリンクをご参照ください。 [Monitor and diagnose Java applications](#)

Java 言語がソフトウェア開発の主流になったことから、Java 仮想マシン (VM) の信頼性は非常に重要な問題となっています。VM は一般的に信頼性の高いソフトウェアですが、実行中にさまざまな理由で障害が発生することもあります。このような障害には VM 自体のエラーによって発生するものもわずかにありますが、大多数は、VM 上のソフトウェア・スタック (例えば、IBM WebSphere Application Server 内) やアプリケーション自体のエラーや誤った構成が原因で発生します。

一般的なプロジェクトでのソフトウェア・スタックは、情報技術が成熟するのに伴い複雑さが増しており、その結果、開発者が問題の原因を判別する上での難しさも増大しています。このように複雑な環境では、障害を診断するための膨大な情報を前にして、途方に迷ってしまう場合もあります。本番環境に至っては、何ギガバイトにも及ぶヒープや何百ものスレッド、数千ものクラス・ローダー、何万ものクラス、そして大量のオブジェクトが存在する可能性さえあります。

拡張可能な IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer (以降、Dump Analyzer と略称) は、そんなジレンマから抜け出す道を探し出すフレームワークです。Dump Analyzer は、IBM Developer Kit for the Java Platform (IBM SDK) を使用して問題を調査する内部の IBM ユーザーおよび外部カスタマーすべてが利用することができます。このツールは複数のアナライザーを使用してフォーマット済みシステム・ダンプを調査し (それぞれのアナライザーがシステム・ダンプで特定の事項を調査)、結果をまとめてスクリプトにリンクさせて簡潔な分析レポートを生成します。Dump Analyzer の最初の 2 つのリリースでは、以下の 4 つの結果のうちのいずれかがレポートされます。

この連載について

連載「[Java の診断を IBM スタイルで](#)」では、Java アプリケーションの問題解決を支援し、アプリケーションのパフォーマンスを改善する新たな IBM のツールを取り上げます。毎回掲載される記事から、すぐに実行に移せる新しい知識を得られるはずです。

この連載の寄稿者はいずれも、Java アプリケーションでの問題解決支援ツールを作成するために新しく組織されたチームのメンバーです。さまざまなバックグラウンドを持つ著者たちが、さまざまなスキルとその専門の分野でチームに貢献しています。

記事に関するご意見・ご質問は、それぞれの記事を担当する著者にお寄せください。

- メモリー不足
- デッドロック検出
- シグナルによる VM 終了 (内部またはミドルウェア/Java アプリケーション・エラーの結果)
- 要詳細調査

最初の 3 つの結果はそれぞれ、次のセクションで説明する VM の問題のタイプのうちの 1 つに対応します。

この記事を読み進めていく上で必要な予備知識はありません。ここでは Dump Analyzer を使ってシステム・ダンプを分析する手順をステップバイステップで説明するとともに、このツールとそのアーキテクチャーについての基礎を概説します。この記事を読み終えれば、Dump Analyzer が

必要になる状況について十分に理解できているはずであり、さらにそのインフラストラクチャーについてもある程度理解ができているはずです。

VM の問題のタイプについての概要

実行中の VM で障害が発生する原因はさまざまにあり、障害のタイプによって異なる診断手法が必要となります。そのため、Dump Analyzer の機能を詳しく検討する前に、さまざまな問題のタイプ、そして問題を解決する上で必要な分析内容について説明しておく価値があります。

メモリー不足の問題

VM はメモリーが不足すると障害が発生します。この場合のメモリーとは、Java ヒープ・メモリーであることも、スレッド・スタック、クラス情報、JIT で生成されたコード、グラフィック要素、そしてベースとなるオペレーティング・システムとインターフェースを取るための他の成果物などを保持するために VM が使用するネイティブ・メモリーであることもあります。

障害を引き起こしたメモリー割り当て自体が根本的な問題の原因になることはまずないため、メモリー不足の問題を診断するのは極めて困難です。原因としては、VM で利用可能なヒープ空間の限界に達するまで、何らかの大規模なコレクションが止め処なく増大し続けたことが考えられます。そのため通常は、ヒープの中身を調べ、さまざまな時点で取得したヒープのスナップショットと比較して、急激に増大したコレクションを特定する必要があります。

デッドロック

デッドロックとは、別のプロセスがリソースを解放するのを待機しているプロセスが複数ある状態のことです。リソース (モニターなど) を所有するスレッドが別のリソースの所有権を取得できない理由は、その取得しようとしている別のリソースを所有するスレッドも同じく、前者のスレッドが所有するリソースの所有権を獲得しようしているためです。このような障害は、パフォーマンス上の問題として現れます。この場合、スレッドの状態とスレッドが所有するリソースを調べることで比較的簡単に診断することができます。

内部エラー

内部エラーは、以下のようにさまざまな問題によって発生します。

- ネイティブ・コードが、無効な入力 (失効したローカル参照など) を持つオブジェクト、または不適切なコーディングをされたオブジェクトにアクセスしようとした場合
- ガーベッジ・コレクターが、参照時に初期化されていないメモリーへのポインターが含まれるようなストレージを誤って再利用した場合
- JIT コンパイラーが、無効なロケーションを参照したり、あるいはそのロケーションに分岐する誤ったコードを生成した場合。

Java アプリケーションまたはミドルウェアでのエラー

Dump Analyzer は現在、VM 自体のレベルで発生または検出したエラーを扱っていますが、この同じ一連のツールはいずれ、VM で動作する Java アプリケーションやミドルウェアでの各種エラーや誤った振る舞いも診断できるようになる予定です。これらのエラーにはさまざまな原因が考え

られますが、通常はアプリケーションまたはミドルウェアのコードに含まれる欠陥、あるいは JVM オプションの誤った使い方や誤った構成によるものです。このようなエラーは一般的に、アプリケーションまたはミドルウェアにおける各種データ構造の状態を調べ、その状態に何らかの点で誤りがないかどうかを判断することによって診断されます。

問題診断の現状

Dump Analyzer のようなツールを使用しない場合、一般的な問題診断プロセスではまず、障害が発生した時点で VM が生成した成果物を調べるところから始めます。これらの成果物には以下があります。

- プロセス空間のダンプ (システム・ダンプまたはコア・ファイル)
- Java ヒープのダンプ (heapdump)
- Java プロセスのスナップショット (javacore ファイル)
- 実行履歴の一部を示すトレース・ファイル

通常、これらの成果物は、特定のフォーマットそれぞれに合わせてカスタマイズしたプログラムを使用して個別に調べることになります。つまり、問題判別のプロセスは基本的に、利用できる情報を手作業で調査することによって進められるということです。このようなプロセスにかかる時間はデータの量が増えるにつれ長くなり、問題判別はますます専門的なジョブとなります。その結果、カスタマーは自分たちで分析を引き受けるのを嫌がり、VM やミドルウェアのベンダーに頼ることが多くなります。しかし、レポートされる問題の大多数は、VM またはミドルウェア自体でのコード変更が必要のないアプリケーション、構成、または環境の問題として診断されるのがおちです。一方で、カスタマーが使用できる診断機能によって、コードの変更を必要とする正真正銘の欠陥だけを VM やミドルウェア・ベンダーに確実にレポートされるようにし、その他の問題については、VM によって生成された適切な成果物を使って、自動化されたプロセスで診断するというのが理想です。

Dump Analyzer の概要

Dump Analyzer は、システム・ダンプを分析して各種の問題を発見するために設計された DTFJ (Diagnostic Tooling Framework for Java。この記事で説明します) をベースとしたツールです。このツールは複数の小さな分析モジュールで構成され、これらのモジュールがそれぞれ特定のダンプ・データを検討し、具体的な問題 (デッドロックなど) の有無を判断します。この設計は新規機能の追加にも簡単に対応可能で、特定の問題を検出するように調整することもできます。

ツールは次の 2 つのレベルで動作します。

- カスタマイズされたそれぞれの分析モジュールがある特定のタイプの問題を診断し、検出された問題についての簡潔な記述を生成します。
- 診断不可能な場合、各分析モジュールはある特定の側面から見たシステムの状態に関する詳細なレポートを生成します。このレポートは、トラブルシューティングの専門家が (おそらく他の情報と併せて) 問題を診断するために使用することができます。

柔軟性をさらに高めるため、分析のフロー制御には単純なスクリプト言語が使用されています。私たちチームは多種多様なスクリプトを用意して、この機能をさらに開発していくつもりです。

このツールの分析フローは以下のとおりです。

1. ツールがユーザーによって選択されたダンプ・データをロードし、詳細分析用のイメージを作成します。
2. ユーザーが、作成されたイメージに対して実行する 1 つ以上の分析モジュールを選択します。ユーザーが特定のアナライザーを選択しない場合は、デフォルト・スクリプトが実行されます。
3. 分析モジュールを実行します。
4. それぞれのモジュールが、詳細分析のフローを左右する情報を返すか、あるいはレポート用の情報を生成します。
5. すべてのモジュールの実行が完了すると、レポートが HTML またはテキスト文書にフォーマット設定されます。

上述のとおり、ユーザーが特定の分析モジュールを要求しない場合はデフォルト・スクリプト (general.sml) が実行されます。このスクリプトは、複数の一般的なタイプの問題があるかどうかを調べる一連のアナライザーを実行します。該当する問題が 1 つも検出されなければ、スクリプトはデフォルト・レポートを呼び出します。このレポートは、ダンプが生成された時点での VM の状態に関する一般情報を要約したものです。

この記事では後ほど、Dump Analyzer の使用例を紹介します。そのなかで、デフォルト以外の選択可能な分析モジュールをいくつか抜粋して概要を説明します。

Dump Analyzer を使用するために必要なセットアップ

Dump Analyzer を実行するために必要となるのは、フォーマット済みシステム・ダンプだけです。システム・ダンプは VM が異常終了するとデフォルトで生成されますが、このようなダンプを別の障害状態にあるときや、ユーザーの要求時に生成するように VM を構成することもできます (詳細は、[「参考文献」](#)に記載した Diagnostic Guide のリンクを参照してください)。

システム・ダンプをフォーマット設定するには、ダンプに対して jextract ツールを実行します。該当するダンプを生成した同じマシン上で同じ VM を使って以下のコマンドラインを実行してください。

```
jextract "corefilename"
```

1.4.2 レベルの VM では、このコマンドによって .sdff ファイルが生成され、Version 5.0 以降の VM では .dmp.zip ファイルが生成されます。オペレーティング・システム・レベルで生成されるダンプのフォーマットを制御するオプションはプラットフォームごとに異なることにも注意してください。特に、これらのオプションが原因でシステム・ダンプが切り捨てられてしまうと、Dump Analyzer が有益な診断を生成できなくなってしまいます。もっともよくありがちなエラー (UNIX システムの場合) は `ulimit` を `unlimited` に設定し忘れることです。他にもプラットフォームごとに重要なオプションがあります。この種の問題を防ぐには、IBM Diagnostics Guide に記載されている情報を参照するか、または IBM Software Support Web サイトで「truncated core」などのキーワードを使ってプラットフォーム固有の技術情報を検索してください (どちらのリンクも「[参考文献](#)」に記載)。

IBM Support Assistant 内で Dump Analyzer を使用する

IBM Support Assistant について

ISA (IBM Support Assistant) は、IBM ソフトウェア製品での問題の解決を支援する無料のソフトウェア保守サービス・ワークベンチです。ISA には、大量の IBM ドキュメンテーションを網羅し、結果をカテゴリー別に分類して検討できるようにする検索機能が備わっています。

ISA には、製品サポート・ページやホーム・ページ、トラブルシューティング・ガイド、そしてフォーラムやニュースグループなどにリンクした製品情報機能もあります。さらに、ISA のサービス機能を使えばデスクトップからの情報を収集し、IBM への問題レポートを簡単に作成することができます。

ISA のツール・ワークベンチには、IBM 製品での問題の解決に役立つ問題判別ツールが用意されています。これらのツールは定期的に更新され、デスクトップでトラブルシューティング・ツールおよび診断ツールを実行できるようにします。ISA のダウンロード・リンクについては、「[参考文献](#)」を参照してください。

Dump Analyzer の主なリリース手段は IBM Support Assistant (以降、ISA とします) です。ISA はすべての内部 IBM ユーザーと外部カスタマーが利用することができます (ダウンロード・リンクについては「[参考文献](#)」を参照)。

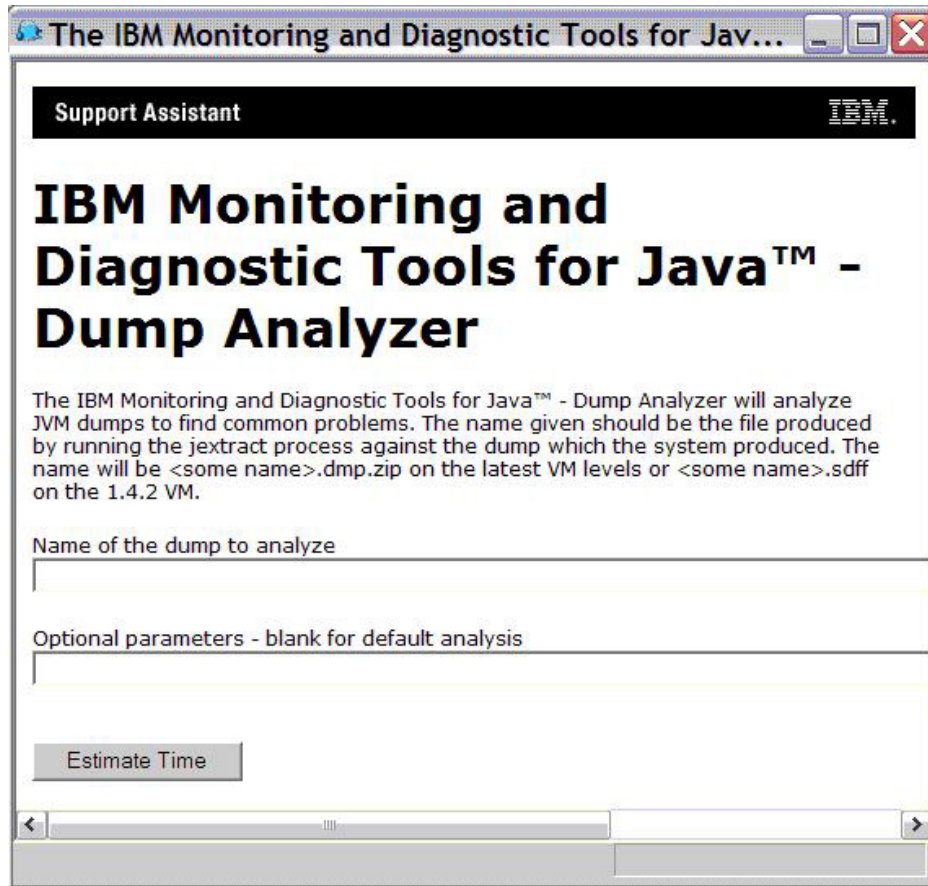
Dump Analyzer を ISA でインストールする手順は以下のとおりです。

1. バージョン 3 の ISA がインストール済みであることを確認します。
2. Dump Analyzer をインストールするには、これに関連する製品プラグイン、例えば IBM Developer Kit for Java をインストールする必要があります (手順については「[参考文献](#)」を参照)。
3. ISA クライアントを再起動します。これで、ツール・プラグインをインストールする準備は完了です。
4. Updater サービスに進みます。それには以下の 2 つの方法があります。
 - Welcome ページで Updater アイコンをクリックします。
 - メニュー・バーにある Updater リンクをクリックします。
5. New Plug-ins タブを選択し、ISA がインストール可能なプラグインのカタログを作成するまで待ちます。
6. Common Component Tools フォルダーを開きます。
7. IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer を選択してインストールします。

Dump Analyzer のインストールが完了すると、ISA 内でこのツールを起動できるようになります。起動方法は以下のとおりです。

1. ISA を再起動します。
2. Tools を選択します。
3. Dump Analyzer を使用できる製品 (この例では IBM Developer Kit for Java) を選択します。
4. IBM Monitoring and Diagnostic Tools for Java - Dump Analyzer をクリックしてツールを起動します。すると、図 1 のような画面が表示されるはずです。

図 1. ISA 内での Dump Analyzer

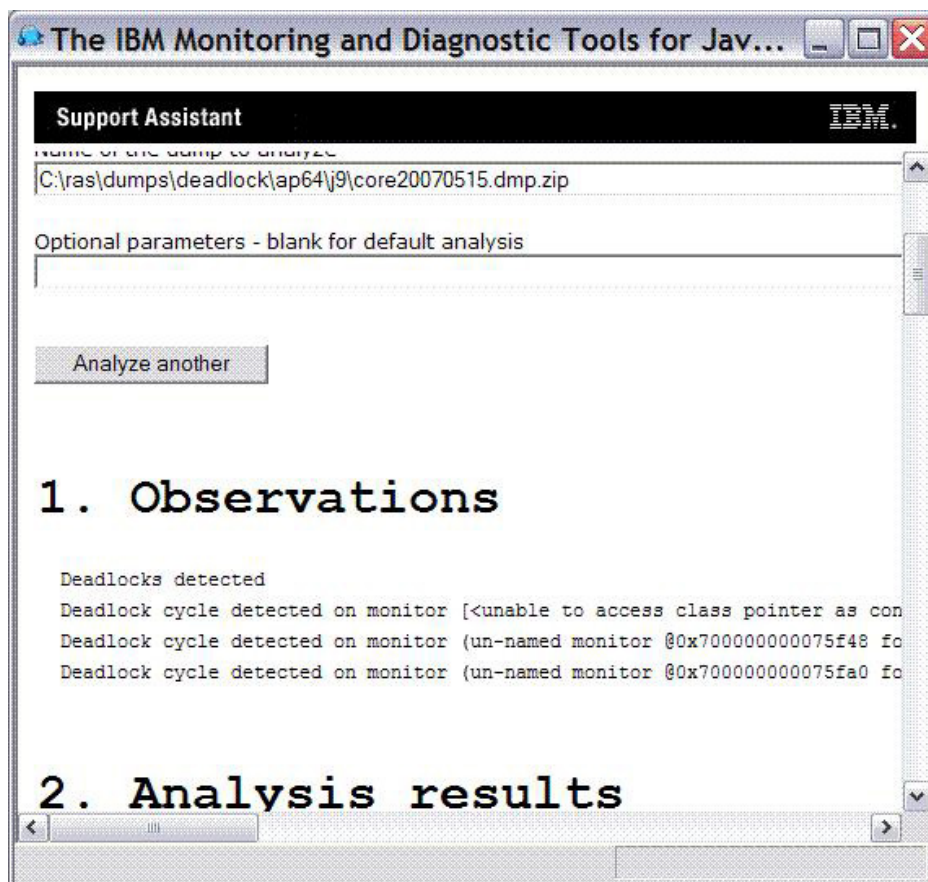


フォーマット済みシステム・ダンプを分析する手順は以下のとおりです。

1. 分析対象のフォーマット済みシステム・ダンプの完全修飾名を入力します。
2. Estimate Time をクリックして、分析にかかる大体の時間を確認します。
3. Analyze をクリックします。分析が完了すると、結果がウィンドウに表示されます。

図 2 は、Dump Analyzer が生成する観測結果の概要の一例です。

図 2. 観測結果の概要の例



Analyze another をクリックすると、図 1 の画面に戻ります。この場合、画面上の最初のテキスト・ボックスは、前に入力したダンプ・ファイルの名前がそのまま入力された状態になっています。

分析モジュールの選択

図 1 と図 2 の起動画面に表示された Optional parameters というラベルが付いたフィールドは、実行されるアナライザーのセット、そしてその他のランタイム・オプションを制御するためのものです。通常はこのフィールドを空白のままにして、デフォルトの分析スクリプト、general.sml が実行されるようにしてください。このスクリプトは、もっとも一般的なタイプの問題をチェックします。ただし、調査中の問題がどのタイプであるかがわかっている場合、あるいはデフォルト・スクリプトに統合されていない問題に対処しなければならない場合には、1 つ以上のアナライザーを明示的に指定して起動することもできます。起動するアナライザーを指定するには、特定のスクリプト・ファイルの名前、または特定のアナライザー・モジュールのクラス名を使います。Optional Parameters フィールドに `-help` と入力すると、詳細なランタイム・オプションがリストされます。

このツールの最初のリリースでデフォルト・スクリプトの他に用意されているのは、少数の試験的なアナライザーだけです。これらのアナライザーには、以下のものがあります。

- DefaultDumpReport (クラス名: `com.ibm.dtfj.analyzer.deal.basic.DefaultDumpReport`): このアナライザーは、主要なすべての側面から VM の状態に関するかなり詳細なレポートを生成

します。このレポートの内容は Javacore ファイルに含まれるようなものです (ただし、DTFJ 固有の情報が追加されています)。

- ListZipJars (クラス名: `com.ibm.dtfj.analyzer.deal.extended.ListZipJars`): この試験的なアナライザーは、VM 内で開かれている現行の zip および JAR ファイルをすべて検出し、それによってアプリケーションまたはミドルウェアが使用中のカスタム・ライブラリーを理解する手掛かりを提供します。
- SystemProperties (クラス名: `com.ibm.dtfj.analyzer.deal.extended.SystemProperties`): この試験的なアナライザーは VM をスキャンし、その VM に定義されているあらゆる Java システム・プロパティーの現行値を出力します。
- WASBasicInfo (クラス名: `com.ibm.dtfj.analyzer.deal.was.WASBasicInfo`): これは極めて基本的かつ試験的なアナライザーで、VM 内で実行する WebSphere Application Server ランタイムの状態をこのツールを使って検査する方法を実演します。

上記のデフォルト以外のアナライザーは、現在、ツールの柔軟性を示すことを主な目的として提供されています。今後のリリースでは、この他にも多くの特殊化されたアナライザーがドキュメンテーションとともにリリースされる予定です。さらにこの連載の第 4 回では、ツールに付属のアナライザーを補完するために独自のアナライザーを作成する方法を紹介します。

コマンドラインから Dump Analyzer を使用する

場合によっては、Dump Analyzer をコマンドラインから実行できると都合がいいこともあります (例えば、既存の問題処理ワークフローに分析を組み込む必要がある場合など)。Dump Analyzer を使用するのにもっとも簡単な方法は、ISA を使うことです。ISA 内に Dump Analyzer をダウンロードする方法は上記ですでに説明しました。

一方、Dump Analyzer を単独で実行するには、以下にリストする 4 つの JAR ファイルと 1 つのスクリプト・ファイルが必要になります。

- `dumpAnalyzer.jar` (`installDir/plugins/com.ibm.java.diagnostics.dbda.isa_(バージョン番号)/WEB-INF/lib` 内)
- `dtfj-interface.jar` (`installDir/plugins/com.ibm.java.diagnostics.dbda.isa_(バージョン番号)/WEB-INF/lib/j9` 内)
- Java 5.0 以降に対応する `dtfj.jar` (`installDir/plugins/com.ibm.java.diagnostics.dbda.isa_(バージョン番号)/WEB-INF/lib/j9` 内)
- Java 1.4.2 対応の `dtfj.jar` (`installDir/plugins/com.ibm.java.diagnostics.dbda.isa_(バージョン番号)/WEB-INF/lib/sov` 内)
- `general.sml` (`installDir/plugins/com.ibm.java.diagnostics.dbda.isa_(バージョン番号)` 内)

上記のファイル・パスではいずれも、`installDir` は ISA インストール・ディレクトリーを表します。このディレクトリーはデフォルトでは、`C:\Program Files\IBM\IBM Support Assistant v3` (Microsoft Windows の場合) または `/opt/IBM/IBM Support Assistant v3` (Linux の場合) となります。これらのファイルは他の場所にコピーすることも、あるいは `installDir/plugins/com.ibm.java.diagnostics.dbda.isa_(バージョン番号)` ディレクトリーにそのまま残し、ここから直接 Dump Analyzer を実行することもできます。ISA は Windows および Linux 上でしか使用できませんが、Dump Analyzer はどのプラットフォームのコマンドラインからでも実行できます。

以下は、Windows のデフォルト・ディレクトリーから Dump Analyzer を実行する場合のコマンドラインのステップです。

1. set CP=WEB-INF/lib/dumpAnalyzer.jar
2. set BCP=WEB-INF/lib/j9/dtfj.jar;WEB-INF/lib/j9/dtfj-interface.jar;WEB-INF/lib/sov/dtfj.jar
3. java -cp %CP% -Xbootclasspath/p:%BCP% com.ibm.dtfj.analyzer.base.DumpAnalyzer (dumpName) (options)

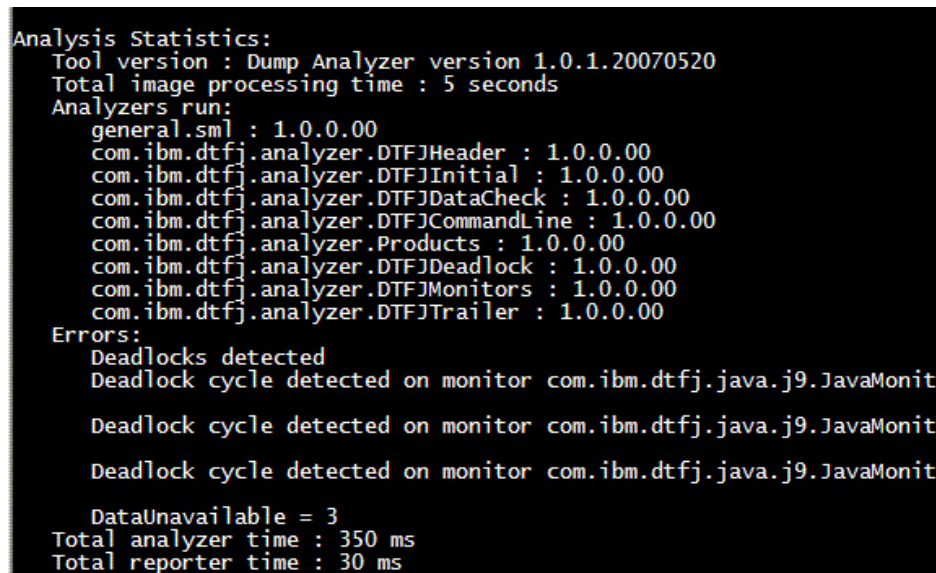
Linux の場合は、以下のステップとなります。

1. export CP=WEB-INF/lib/dumpAnalyzer.jar
2. export BCP=WEB-INF/lib/j9/dtfj.jar:WEB-INF/lib/j9/dtfj-interface.jar:WEB-INF/lib/sov/dtfj.jar
3. java -cp \$CP -Xbootclasspath/p:\$BCP com.ibm.dtfj.analyzer.base.DumpAnalyzer (dumpName) (options)

ここで、dumpName は分析するダンプの完全修飾名で、options は Dump Analyzer を構成するために使用できるランタイム・パラメーターです。-help オプションを指定して実行すると、使用可能なすべてのパラメーターのリストが出力されます。

図 3 のスナップショットは、コマンドラインで実行中の Dump Analyzer からの出力です。

図 3. Dump Analyzer のコマンドライン出力例



```
Analysis Statistics:
Tool version : Dump Analyzer version 1.0.1.20070520
Total image processing time : 5 seconds
Analyzers run:
  general.sml : 1.0.0.00
  com.ibm.dtfj.analyzer.DTFJHeader : 1.0.0.00
  com.ibm.dtfj.analyzer.DTFJInitial : 1.0.0.00
  com.ibm.dtfj.analyzer.DTFJDataCheck : 1.0.0.00
  com.ibm.dtfj.analyzer.DTFJCommandLine : 1.0.0.00
  com.ibm.dtfj.analyzer.Products : 1.0.0.00
  com.ibm.dtfj.analyzer.DTFJDeadlock : 1.0.0.00
  com.ibm.dtfj.analyzer.DTFJMonitors : 1.0.0.00
  com.ibm.dtfj.analyzer.DTFJTrailer : 1.0.0.00
Errors:
  Deadlocks detected
  Deadlock cycle detected on monitor com.ibm.dtfj.java.j9.JavaMonit
  Deadlock cycle detected on monitor com.ibm.dtfj.java.j9.JavaMonit
  Deadlock cycle detected on monitor com.ibm.dtfj.java.j9.JavaMonit
  DataUnavailable = 3
  Total analyzer time : 350 ms
  Total reporter time : 30 ms
```

DTFJ についての詳細は、「[参考文献](#)」セクションを参照してください。

今後の予定

この記事を書いている時点で利用可能になっているのは Dump Analyzer の初期リリースですが、私たちチームは今後も定期的に機能拡張と更新を行っていく予定です。特に重点を置く分野としては、以下の 2 つがあります。

DTFJ: Dump Analyzer を支えるアーキテクチャー

Dump Analyzer は、DTFJ を使用してフォーマット済みシステム・ダンプを調べます。DTFJ は Java 診断ツールの作成をサポートする API で、VM から取得したシステム・ダンプを検査することができます。システム・ダンプを検査できるようにするには、まず jextract で処理して Java ランタイムに固有の情報を追加しなければなりません。jextract をシステム・ダンプに対して実行(コマンドラインで `jextract core.dmp` を実行) する際に必須なのは、Java プラットフォームと同じバージョンを使用すること、そしてシステム・ダンプを生成したマシンで実行することです。このようにして実行して生成されたファイル (Java プラットフォームが V1.4 の場合は `.sdff`、V5.0 以降の場合は `.dmp.zip`) は、どのシステムでも検査することができます。

DTFJ による分析での第一段階は、適切なイメージ・ファクトリーを使用して DTFJ イメージを作成することです。このファクトリーは特定の VM レベルに固有なもので、プラットフォーム特有のダンプ・フォーマットを認識します。DTFJ イメージはオペレーティング・システム全体のイメージを表し、システム・ダンプを生成したマシンのアーキテクチャーに関する情報にアクセスするためのメソッドを提供しますが、その主要な機能はイメージに含まれるアドレス空間の検査を可能にすることです。DTFJ API を使用すれば、イメージからランタイム・パラメーター、Java スレッド、ネイティブ環境、そしてヒープ上のオブジェクトの詳細にナビゲートすることができます。このインターフェースによって、ツールによる問題の分析とレポート情報の生成が可能になります。

- 引き続きツール自体のユーザー・インターフェースを強化し、ダンプおよび実行するアナライザーを制御するためのパネルの追加、出力フォーマットの改善、そしてできれば対話モードの追加などを行います。
- アナライザーとスクリプトの数を増やし、より広範な問題に対応できるようにします。

特に期待できるのは、新しいアナライザーを作成するという部分です。この DTFJ ダンプ分析技術により、極めてアクセスしやすいメカニズムでスレッドやモニターなどの低レベルの VM エンティティーを検査し、メモリー不足のエラーや異常終了、デッドロックなどの問題を診断できるようになっています。さらに、VM に存在するあらゆるデータ構造のコンテンツ (具体的には、VM 内で動作するアプリケーションまたはミドルウェアの実装に含まれる各種データ構造のコンテンツ) も調べることができます。私たちは、この情報を利用する一連のアナライザーの作成に着手し、WebSphere Application Server、そしてできれば他のスタック製品でのさまざまな問題の診断に役立てたいと思っています。

チームの目的は、このツールをできる限り有益なものにすることなので、ツール自体に関することでも、新しく追加したいアナライザーに関することでも、フィードバックは大歓迎です。フィードバックは、この記事または ISA から投稿することができます。

連載の今後の予告

連載の次回の記事では、IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer を紹介します。このツールは、詳細なガーベッジ・コレクションのログを分析することにより、メモリー・ベースの Java パフォーマンス問題の調査を支援します。このツールを使用すると、メモリー使用量のパターンを検討したり、メモリー・リークの有無を判断したり、あるいはパフォーマンスを改善するためにガーベッジ・コレクションの構成を調整したりすることができます。

Dump Analyzer については、連載第 4 回でもう一度取り上げます。その際に、このツールの拡張性をさらに深く掘り下げ、独自の分析モジュールを作成する方法を説明します。

著者について

Helen Beeken



Helen Beeken 博士は、RAS ツールの開発に従事する IBM Java Technology Centre のソフトウェア開発者です。本チームに加わる前は、オープン・ソースの AspectJ および AJDT Eclipse プロジェクトに携わり、大規模なソフトウェア・システムの負荷試験を支援していました。

Daniel Julin



Daniel Julin は、複合オンライン・システムの開発およびトラブルシューティングに 20 年の経験を積んでいます。WebSphere Serviceability Team の技術面でのリーダーとして、現在は主に WebSphere Application Server での問題判別を支援し、IBM サポートの効果を最大限に生かす一連のツールおよび技術を定義、実装する上でのチームの活動を手伝っています。また、さまざまな局面で重要なカスタマー・サポートに直接関わることもあります。

Julie Stalley



Julie Stalley は、現在 Hursley に置かれた Java RAS Tooling チームのメンバーとして活躍するソフトウェア開発者です。1996年に IBM に入社してから 5 年間、大規模なクライアント/サーバー・アプリケーションの開発に携わった後、2001年に Hursley に移りました。それ以来、Java クラス・ライブラリーの入出力、ネットワーキング、および XML を専門に取り組んでいます。

Martin Trotter



Martin Trotter は Java プラットフォームをその初期の頃から扱ってきました。特に Java VM とガーベッジ・コレクターについては広範な経験を持ちます。現在は、Java VM での問題の診断に有効なツールの改善に取り組んでいます。

© Copyright IBM Corporation 2007, 2013

(www.ibm.com/legal/copytrade.shtml)

商標

(www.ibm.com/developerworks/jp/ibm/trademarks/)