Oracle JRockit Mission Control の概要

Oracle ホワイト・ペーパー
2008 年 6 月

ORACLE

Oracle JRockit Mission Control の概要

Oracle Corporation 発行「Oracle JRockit Mission Control Overview」の翻訳版です。

Oracle JRockit Mission Control の概要

JRockit Mission Control で提供されるスムーズな JVM のプロファイリングおよび診断は、本番環境での使用に適しています。

はじめに

Oracle JRockit Mission Control は、Oracle JRockit JVM 上で動作する強力なツール・セットです。これらのツールでは、開発環境および本番環境の両方での使用に適した、高度でスムーズな JVM の監視および管理機能が提供されます。ここでは、JRockit Mission Control を紹介し、製品のおもなコンポーネント、この製品のコンポーネントと競合するテクノロジーとの差異、また、JRockit JVM 上で実行した場合の、アプリケーションを監視、管理、プロファイリング、診断する機能の使用方法について説明します。

スムーズなプロファイリングおよび診断

現在、JRockit Mission Control には、次の 3 つの主要なツールが含まれています。

- Console
  JMX ベースの管理コンソール
- JRA
  フライト・レコーダーのように機能するメイン・プロファイラ
- Memleak
  メモリ・リークを検出して解消するためのツール

Java ランタイムの監視、管理、およびプロファイリングに現在使用されているテクノロジーのほとんどは、バイト・コードの埋込みおよび（旧版の JVMPI と置き換えられた）JVMTI のような、パフォーマンスに大きな影響を与えるものが利用されています。JRockit Mission Control のおもな目的は、実行されているシステムにできるだけ影響を与えずに必要なデータを収集することです。また、使用されているテクノロジーは、ツールを JVM から切り離すことで、アプリケーションの最高速度での実行を可能にします。そのため、JRockit Mission Control は本番システムでの使用に適しています。さらに、ごくわずかなオーバーヘッドしか発生しないので、ハイゼンベルク効果を最小限に抑え、オーバーヘッドが大きくなりがちな手法よりもアプリケーションをより的確に表すデータを提供できます。

JRockit Mission Control には、次の 3 つの強力なツールが含まれています。

JRockit Management Console

JRockit Management Console は、複数の JRockit インスタンスを監視および管理するツールです。このツールを使用することで、ガベージ・コレクションによる休止時間、メモリと CPU の使用状況に関する実データ、および JVM の内部 MBean サーバーに配置された JMX MBean の情報を取得し、表示できます。JVM の管理には、CPU アフィニティ、ガベージ・コレクション方式、メモリ・プールのサイズなどに対する動的な制御が含まれます。

JRockit Runtime Analyzer

JRockit Runtime Analyzer（JRA）は、JVM および JVM で実行しているアプリケーションに関する詳細情報の記録を生成するオンデマンドの'フライト・レコーダー'です。記録されたプロファイルは、あとから JRA Mission Control プラグインを使用してオフラインで分析できます。記録されるデータには、メソッドとロックのプロファイリングのほかに、ガベージ・コレクションの統計、最適化の判断、オブジェクト統計、latency イベントなどがあります。

JRockit Memory Leak Detector

JRockit Memory Leak Detector は、メモリ・リークを検出してその原因を突き止めるためのツールです。JRockit Memory Leak Detector では、傾向分析によって、非常にゆっくりと進行するメモリ・リークを検出できます。Memory Leak Detector には、詳細なヒープの統計（リークが発生しているオブジェクトへの参照のタイプとそのオブジェクトを参照しているインスタンスを含む）と割当て場所が表示されるので、メモリ・リークの原因をすばやく絞り込むことができます。Memory Leak Detector では高度なグラフィカル表示手法が使用されているため、複雑な情

Oracle Corporation 発行「Oracle JRockit Mission Control Overview」の翻訳版です。

報をより簡単に操作して把握できることがあります。

このあとにある 3 つの章では、各ツールについて詳しく説明します。

JRockit Management Console

Management Console は、JMX に準拠している任意の JVM の監視に使用できる JMX コンソールです。

JRockit Management Console は JMX ベースのコンソールで、JRockit JVM の管理および監視に使用します。Management Console では、ライブ・セット、ヒープの使用状況、CPU の負荷などの重要な状態データ、および JVM の内部プラットフォーム MBean サーバーに登録されており、MBean によって公開されるそのほかの属性情報が提供されます。また、Management Console には、オーバーヘッドの小さいオンラインのメソッド・プロファイラおよび例外カウンタも含まれています。

Management Console の機能を十分に活用するには、JRockit JVM を監視している必要があります。

Management Console は、JRockit プロセス内で動作するエージェント（JVM の内部プラットフォーム MBean サーバーに登録されている MBean を公開）と、JRockit Mission Control 内で動作する個別の GUI プラグインで構成されています。Management Console を JRockit JVM に接続すると、JRockit 固有の MBean のセットが自動的に作成されてプラットフォーム MBean サーバーに登録され、JRockit 固有の機能が公開されます。

図 1 に示すように、1 つの Management Console を複数の JRockit JVM に接続でき、また Management Console の複数のインスタンスを 1 つの JRockit JVM に接続することもできます。なお、JRockit Mission Control は複数の JRockit JVM に対処できるので、通常は同一マシン上で JRockit Mission Control の複数のインスタンスを実行する必要がないことに注意してください。

1 つの JRockit Mission Control から、複数の Management Console を異なる JVM に接続できます。

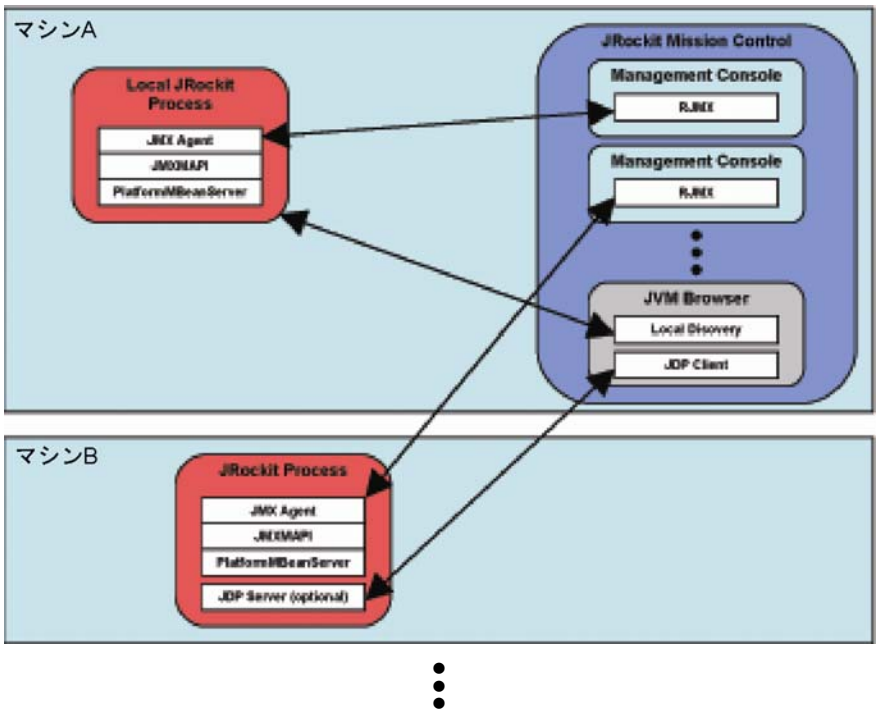

図 1：BEA JRockit Management Console の通信

JRockit JVM には、Management Console に JRockit 固有のデータを公開する独自の MBean、コンソールからリモートでデータにアクセスできるようにするエージェント、および実行されている JRockit を自動検出できるようにするサーバーが含まれています。

高度なアーキテクチャ上の観点から見ると、監視する JRockit の次の部分がコンソールと関係しています。

- **インタフェースのセット（JMXMAPI）**

  JRockit 固有の MBean で、JRockit 固有のデータをほんのわずかなオーバーヘッドで提供します。この MBean の集合には、JRockit のすべてのパフォーマンス・カウンタを属性として公開するために動的に生成された MBean が含まれます。これらの拡張機能は、内部的に JRockit JMX Management API、または省略形で JMXMAPI と呼ばれています。

- **JMXMAPI を公開するエージェント**

  コンソールとの通信には、リモート JMX over RMI が使用されます。エージェントは、JRockit JVM のコマンドライン・パラメータを使用して起動できます。また、エージェントのライフ・サイクルは、さまざまな JRockit 固有の API および jrcmd によって制御できます。jrcmd は JRockit JVM 用のコマンドライン・ユーティリティであり、Oracle JRockit JDK で提供されています。

- **JDP（JRockit Discovery Protocol）サーバー**

  Management Console では、マルチキャストを使用して特定の JRockit の位置情報を取得する方法で、ネットワーク上で動作している JRockit JVM の自動検出がおこなわれます。JDP サーバーはオプションですが、JRockit JVM のコマンドライン・パラメータまたは jrcmd を使用して起動できます。

バージョン 1.0 を除き、JRockit Mission Control のすべてのバージョンにおいて、JDP は JVM ブラウザの機能拡張であり、Management Console の一部ではないことに注意してください。

コンソールではオンラインの正確なメソッド・プロファイラもサポートされており、呼出し回数およびメソッドのタイミング情報が提供されます。ただし、JRockit Mission Control で推奨されるプロファイリングは、JRockit Runtime Analyzer を使用する方法です。

コンソール・クライアント・プラグインでは、JMX サービス・レイヤーがサポートされています。このレイヤーでは、永続性、属性サブスクリプションと通知、JVM の自動検出、および Management Console 用の新しいプラグインを記述するための拡張可能なフレームワークが提供されます。

高度なアーキテクチャ上の観点から見ると、Management Console クライアント側には次のプラグインがあります。

- **RJMX（JRockit Remote JMX サービス）**

  永続性、属性サブスクリプションの抽象化、通知フレームワークなどのサービスを提供します。

- **JDP クライアント**

  JDP が有効になっている JRockit JVM を自動的に検出します。

  （繰り返しますが、これが有効なのは、JRockit Mission Control のバージョン 1.0 のみです。最近のバージョンでは、JDP クライアントは JVM ブラウザの拡張機能となっています。）

- **メイン・コンソール・プラグイン**

  Management Console のフレームワークと、Management Console を拡張するほかのプラグイン用の拡張ポイントを定義します。

- **さまざまなプラグイン**

  Overview、MBean Browser、Method Profiler などのデフォルトのタブを表示する一連のプラグインは、Management Console で確認できます。

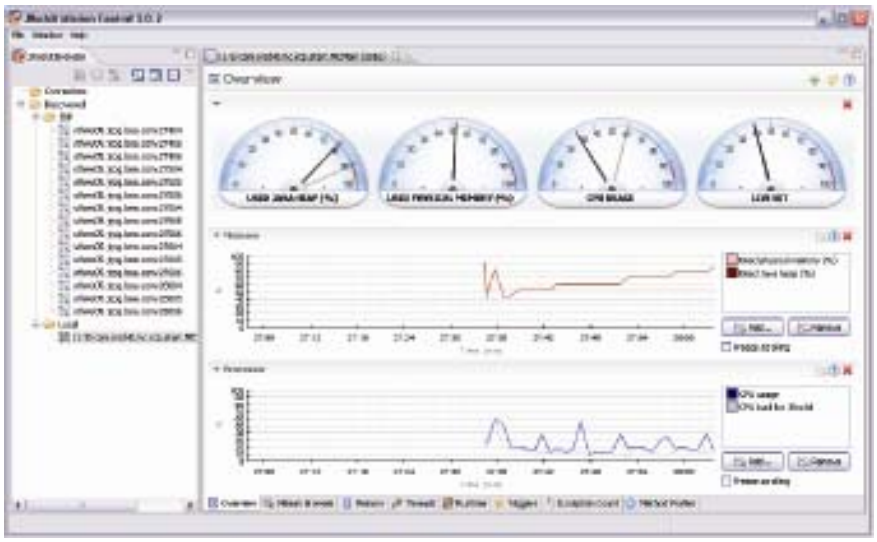

図 2：Management Console の GUI

Management Console アプリケーションには属性サブスクリプションの概念が導入されています。簡単にいえば、これは MBean のオブジェクト名（ObjectName 型）、属性名、およびサブスクリプション間隔で定義されます。コンソールでは、通知ルールの追加や、このような属性サブスクリプションからのデータのグラフ表示と永続化をおこなうことができます。属性サブスクリプションは、通常の JMX MBean 属性、属性がもとになっている個々の複合データ・キー、または JMX 通知データをベースにできます。必要に応じて、データに対応するほかの複数の属性サブスクリプションに基づいた独自の複合属性サブスクリプションを作成したり、合成属性サブスクリプションを作成したりできます。後者の場合、データは実装者が提供します。

メソッド・プロファイラを使用すると、メソッドの呼出し回数とそのメソッドの実行にかかった時間を、ほんのわずかなオーバーヘッドで調べることができます。対象となるメソッドのマシン・コードはわずかな量の計測用コードを付加して再生成されるだけで、そのコードはプロファイラが停止するとただちに削除されます。したがって、メソッド・プロファイラの使用によるオーバーヘッドは、プロファイリングするメソッドを選択したときにのみ発生し、さらに選択したメソッドについてのみ発生します。

しかし、この種のメソッド・プロファイリングが有効なのは調べる対象が事前にわかっている場合のみなので、ほとんどの場合、プロファイリングには JRockit Runtime Analyzer を使用する必要があります。

まとめると、Management Console は、次のような豊富な機能を備えた、柔軟性の高い JMX 管理/監視ツールということになります。

- 任意の数値属性をグラフ表示する機能
- オフライン分析を可能にする、任意の属性セットの永続化
- 休止時間、ライブ・セットのサイズ、および連続的なヒープの使用状況の手軽なグラフ表示を可能にする、特別な属性サブスクリプション
- 特定の属性についてユーザーの指定した条件が発生した場合にアクションを実行できる通知ルール
- 通知アクションと通知制約の独自コードをユーザーがプラグインできるようにする機能
- ガベージ・コレクション戦略、ヒープ・サイズ、ナーサリ・サイズ、JRockit プロセス・アフィニティ、詳細出力フラグの有効化/無効化などの動的な変更を含む、Java ランタイムの管理
- オーバーヘッドの小さいメソッド・プロファイラ
- 例外カウンタ
- MBean の操作の動的な呼出し
- JRockit 診断コマンドの呼出し

SSL、認証、ロールを使用するようにコンソールを構成する方法の詳細については、JRockit のドキュメントを参照してください。

JRA（JRockit Runtime Analyzer）

JRockit Runtime Analyzer は本来、JRockit 開発者にフィードバックを提供するために作成されました。JVM とその JVM で実行する Java アプリケーションの診断、プロファイリング、およびチューニングのために、JRockit サポート部門で内部的に使用され、さらにお客様によっても使用されます。

JRA は、Java アプリケーションと JVM のプロファイラです。JRA は、かなり前から JRockit 開発チーム内で活用されてきたものであり、本来は、実在のアプリケーションに基づいて JVM を最適化する方法を JRockit 開発者が見つけられるように作成されました。そのあと、当社のお客様にとっても、本番環境および開発の両面で問題を解決する際に非常に役立つことがわかりました。また、JRA は、JRockit サポート部門がお客様の問題解決を支援するために使用する主要ツールの 1 つでもあります。

JRA はフライト・レコーダーのように機能し、あらかじめ設定した期間における Java アプリケーションと JVM の動作が記録されます。得られた記録は、あとから JRA Mission Control プラグインを使用して分析できます。たとえば、ホット・メソッドのコール・トレース、不正な同期、およびそのほかの重要なアプリケーション/JVM の動作を分析できます。オラクルのサポート部門では、お客様から定期的に提供していただく JRA の記録を使用して、お客様の問題解決を支援しています。

JRA は、JVM の記録エンジンと、得られた記録の分析に使用できる GUI プラグインのセットの 2 つの部分から構成されています。記録エンジンは、JRockit Hot Spot Detector（最適化エンジンによって、最適化するメソッドの決定にも使用される）、オペレーティング・システム、メモリ管理システム（とくに、ガベージ・コレクタ）、JRockit ロック・プロファイラ（有効になっている場合）など、複数の情報源を使用します。

JRA ツールを利用すると、JRA の記録に含まれている情報をグラフィカルな方法で簡単に分析できます（**図 3** を参照）。記録期間中の休止時間、ヒープの使用状況、コミットされたヒープ・サイズが、グラフに表示されます。個々のガベージ・コレクション（GC）を任意に選択でき、選択したガベージ・コレクションに関する非常に詳細な情報が表示されます。

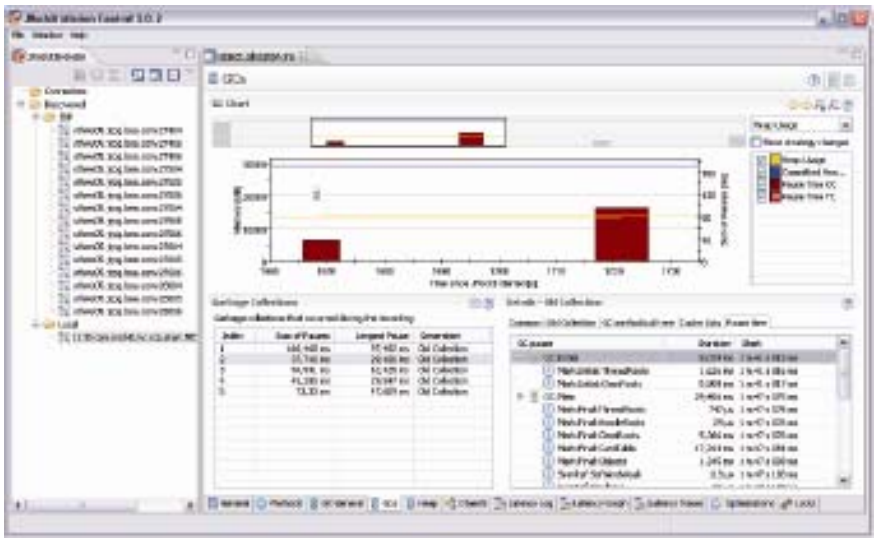

図 3：JRA のガベージ・コレクション情報

JRockit Runtime Analyzer を使用すると、ガベージ・コレクション・プロファイリング、メソッド・プロファイリング、latency イベント、ロック・プロファイリング、ヒープのヒストグラムなどの詳細な情報を取得できます。

また、ホット・メソッドのコール・トレースが表示されますが、その中には、該当するメソッドのコールにつながるコール・トレースだけでなく、その次に通常何がコールされたかを表すコール・トレースも含まれます。さらにコール・トレースでは、該当するメソッドの最適化バージョンがコールされたかどうかも示されます。

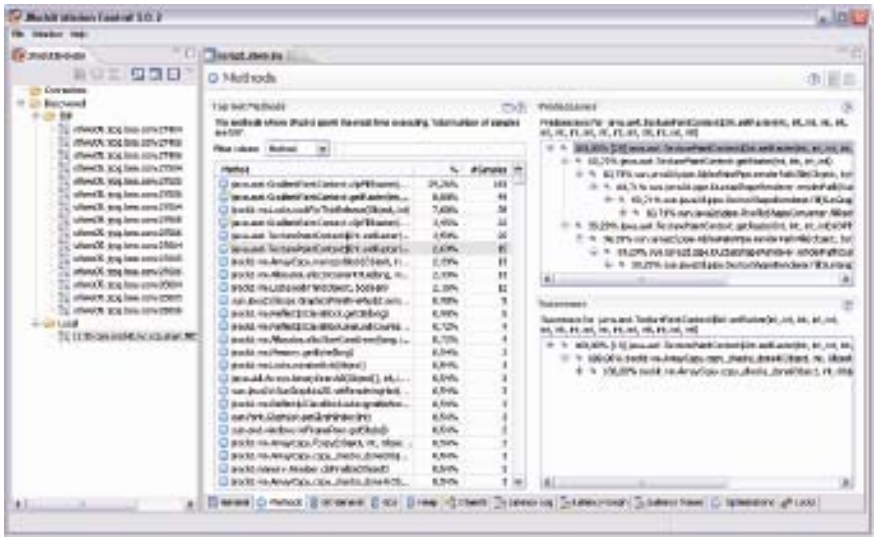

図 4 : JRA のメソッド・プロファイラ

記録の開始時と終了時にヒープのヒストグラムが取得され、使用済みヒープ領域の占有率が 0.5%以上のクラスのインスタンスについて、クラスごとのヒープの使用状況が表示されます。さらに収集されたヒープ情報は、ヒープの使用状況を示す円グラフの描画にも使用されます。

JRockit R27.3 以降では、JRockit Mission Control に、Java の latency をプロファイリングしてグラフィカルに表示できる **latency アナライザ**が含まれています。JRockit Mission Control は、これまでも常に、CPU バウンド問題といった純粋な計算のオーバーヘッドによって発生するパフォーマンスの問題について、その原因を特定する際に有効でした。しかし、R27.3 よりも前では、latency に関連する問題の原因を突き止めるのに適した方法がありませんでした。たとえば、別の理由でスレッドが停止してしまうために、アプリケーションでスループットの問題が発生している場合は、対処できませんでした。

R27.3 と latency 検出ツールの組合せでは、どこでスレッドが停止しているかを示すスレッド・グラフを表示できます。強力で適切に最適化された可視化技術により、数十万のイベントを非常にすばやく可視化、ズーム、およびピックアップすることが可能になりました。また、メソッド・コールのツリーを表示してイベントの発生元を確認すること、またはさまざまなフィルタ機能があるログ・ビューで RAW イベントを表示することも可能です。

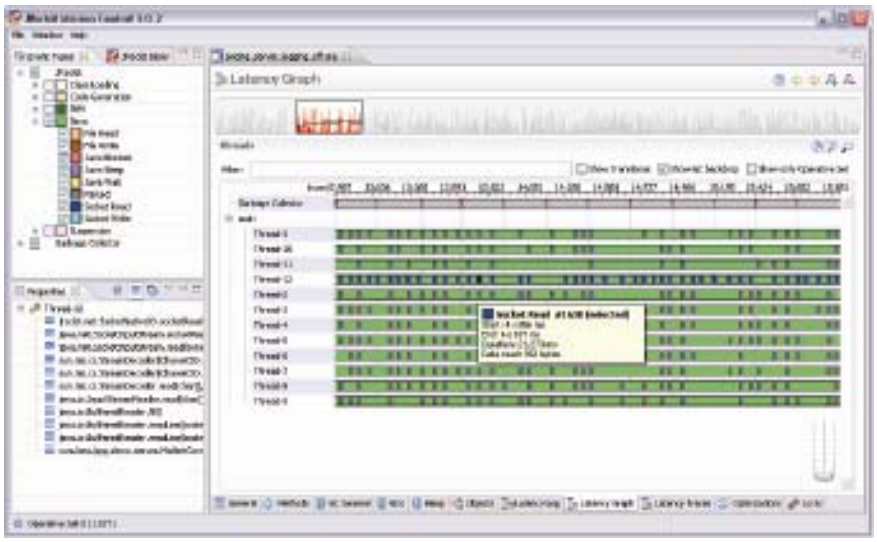

図 5：JRA の Latency ツール

JRA の記録を開始するには、次のようにいくつかの方法があります。

- JRockit Mission Control の GUI を使用
- JRCMD を使用（JRockit JVM 用のコマンドライン・ツール）
- JRockit の拡張機能 MBean を使用
- JRA を使用（XXjra コマンドライン・パラメータ）

JRA の記録開始には、Management Console で通知ルールを使用してトリガすることもできます。たとえば、90%を超える CPU 負荷が 1 分間続いた場合に JRA の記録を開始するように、ルールを作成できます。

以上のことからわかるように、JRA は以下の機能を備えた強力な JVM/Java アプリケーション・プロファイラです。

- JVM と Java アプリケーションの両方を効率的にプロファイリングする機能（通常、オーバーヘッドは 2%未満）
- メソッド・コール・トレースで、対象のメソッドに到達するまでのパスとその次に何がコールされたかを表示する機能
- メソッド・ホット・スポット・テーブルで、どのメソッドがもっとも頻繁にコールされたかを表示する機能
- 非常に詳細なガベージ・コレクション統計情報により、各ガベージ・コレクションの発生時の状況を表示する機能
- ガベージ・コレクション戦略の変更
- ヒープの使用状況および休止時間をグラフ表示する機能
- ヒープのヒストグラムで、記録の開始時と終了時におけるクラスごとのヒープの使用状況を表示する機能
- 詳細なロック・プロファイリングにより、どのロックがかけられたか、それらが競合したかどうか、何回かけられたかを表示する機能
- 断片化を含め、オブジェクト・サイズごとのヒープの使用状況を円グラフで表示する機能
- 記録中に最適化されたメソッドの情報を提供する機能
- Latency プロファイリング

JRA の使用方法の詳細については、JRockit Mission Control のドキュメントを参照してください。

JRockit Memory Leak Detector

Memory Leak Detector では、徐々に進行するメモリ・リークもすばやく検出できます。アプリケーションが最高速度で動作している間も、オンラインで分析がおこなわれます。

JRockit Memory Leak Detector は、メモリ・リークをすばやく検出するのに役立つツールです。Java の自動メモリ管理のおかげで、使用するメモリの割当てと解放を開発者が明示的に指定する必要はなくなりましたが、使用されなくなったオブジェクトをプログラムで参照し続ければ、依然としてメモリ・リークが発生する可能性は残ります。

Memory Leak Detector には、徐々に進行するメモリ・リークもすばやく検出できる傾向分析機能が用意されています。傾向分析機能では、アプリケーションのヒープの使用状況がクラスごとに示されます。各タイプのインスタンスによるヒープ領域の使用量、それらのインスタンスによるヒープ領域の占有率、存在するインスタンスの数、およびそれらの使用量の増加率（バイト/秒単位）が表示されます。

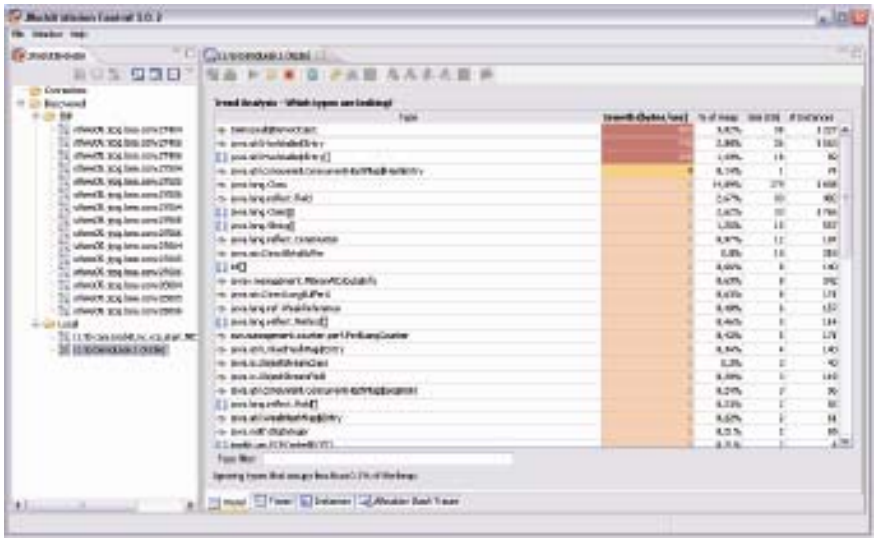

図 6：メモリ・リークの傾向分析テーブル

JRockit Memory Leak Detector では、ヒープ上でのさまざまなタイプの関係や、各タイプとインスタンスの関係が対話形式のグラフに表示されます。

JRockit Memory Leak Detector には、リークの原因をすばやく突き止めるための手段も用意されています。メモリ・リークの疑いがあるタイプを傾向分析テーブルで選択し、参照しているインスタンスが存在するタイプをグラフに表示できます（図 7 を参照）。ユーザーはグラフのノードを任意に展開して、参照関係の起点にまでさかのぼることができます。また、クラスのインスタンスを表示してイントロスペクトし、選択したインスタンスを参照するすべてのインスタンスをインスタンス・グラフに表示できます。さらに、割当てトレースを有効にすることで、特定のクラスの割当てをすべて追跡できます。

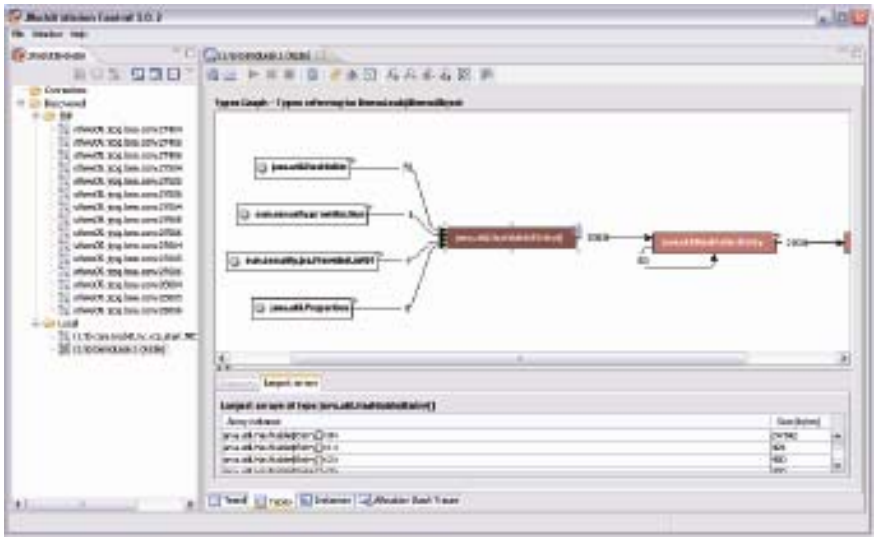

図 7：メモリ・リークのクラス・グラフ

Memory Leak Detector のセッションが開始されると、Memory Leak Server（MLS）が起動します。接続する JRockit JVM の JDK がバージョン 5.0 の場合には JLMEXT の JMX コネクタが、バージョン 1.4 の場合には Rockit Management Protocol（RMP）が、それぞれ使用されます。プロトコルが異なること、および両者のネットワーキング特性とセキュリティには違いがあることに注意してください。MLS はネイティブ・サーバーであり、セッション中における通信の残りはこのサーバーとの間でおこなわれます。クライアント側では、このネイティブ・サーバーとの通信に Java API が使用されます（図 5 を参照）。ネイティブ・サーバーを使用する理由は、深刻なメモリ・リークの状況が発生して JRockit JVM が Java ヒープを使い果たしてしまうと、JRockit はそれ以上 Java コードを実行できなくなるからです。

図 8：Memleak セッションの確立

Memory Leak Detector では、ガベージ・コレクタのマーク・フェーズを巧みに利用して、必要な処理（ガベージの収集）の実行に影響しない少しのブックキーピング処理を追加することで、このパフォーマンスを実現しています。

Memory Leak Detector では、ガベージ・コレクタのマーク・フェーズを巧みに利用していくらかのブックキーピング処理を追加することで、ヒープのヒストグラム（クラスごとに集計されたヒープのメモリ統計情報）の記録および傾向分析の生成をおこないます。競合するほかのツールにおけるメモリ・リークの検出によく使用される方法の 1 つは、システム全体の全ヒープのスナップショットをいくつか作成し、それらを比較することです。数百ギガバイトのヒープ領域があるシステムや、本番環境にあるシステムの場合は、この方法が適さないことがあります。JRockit Memory Leak Detector では、対象の情報のみが回線で送信されるので、必要な帯域幅は非常に小さくなります。割当てコール・トレース機能を使用する場合は、選択したクラスの割当て場所に関係するコードのみがインスツルメンテーションの対象になります。さらに、セッションが終了するか、または割当てトレースが無効になるとすぐに、インスツルメンテーションは削除され、これらの割当て場所に関係するコードは再び最高速度で動作するようになります。

まとめると、JRockit Memory Leak Detector は、次のような多数の新機能を備えた高度な分析ツールということになります。

- 徐々に進行するメモリ・リークも検出可能な、傾向分析を実行できます。
- たとえば、ヒープ全体を何度もクライアントにダンプして違いを調べるのではなく、JRockit メモリ管理システムを巧みに利用して、分析をオンラインで実行します。
- リークを起こしているタイプとそれ以外のタイプとの関係、またはリークを起こしているインスタンスとそれ以外のインスタンスとの関係を見つけて分析できる高度なユーザー・インタフェースを提供します。
- オーバーヘッドがきわめて小さいため、本番環境でのオンライン分析にも使用できます。
- バイト・コードの埋込みは使用されません。Memory Leak Detector で分析する場合、Java コードはあたかも接続されていないかのように実行を続けます。Java コードは一切変更されません。

JRockit Memory Leak Detector の詳細については、JRockit Mission Control のドキュメントを参照してください。

結論

JRockit Mission Control は、Java アプリケーションの監視、管理、プロファイリング、およびメモリ・リークの解消に適した多目的ツールです。開発用途には無料で利用できます。また、信頼性が高く本番環境で使用でき、現在使用されている類似のツールよりもパフォーマンスのオーバーヘッドがはるかに小さく、使用後にはシステムにまったく痕跡を残しません。

Oracle JRockit Mission Control の概要
2008 年 6 月
著者：Marcus Hirt

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問い合わせ窓口：
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200
www oracle.com