3C-4

携帯端末 Java アプリケーションに特化した高速化手法

高橋　克英　　岡田　英明　　清原　良三
三菱電機(株)　情報技術総合研究所

1. はじめに

近年、携帯端末に搭載されている Java 実行環境には、ゲームや株価表示等の様々なアプリケーションが提供されている。株価表示、案内等の情報提供を行うアプリケーションでは、実行速度は問題とならない。しかし、ゲーム、特にシューティング・ゲームでは重要な問題であり、実行速度の向上に強い関心が持たれている。

Java アプリケーションは、Java バイトコード(以下 バイトコード)に変換された上で、Java 仮想マシン(以下 JavaVM)[1]で実行される。JavaVM は、バイトコードをインタプリタ処理で実行することが一般的であり、実行速度が遅い。筆者は、メモリ資源の限られた携帯端末上でバイトコードの実行性能を向上するために、アプリケーションの特性と構造に着目した動的コンパイル方式の検討と評価[2,3]を行ってきた。

JavaVM が行うバイトコードの実行以外に、Java アプリケーションの性能に影響を与える機能として、Java クラスライブラリが提供する機能が挙げられる。本論文では、Java クラスライブラリの中で頻繁に利用される描画と読み込み/書き込み処理の高速化手法の実装と評価を述べる。

2. 動的コンパイル方式による高速化の効果

動的コンパイル方式は、Java バイトコードをネイティブコードに変換することで高速化を実現している。Java クラスライブラリの処理内容の多くは、携帯端末のシステムに組み込まれたネイティブライブラリを用いて実装されているため、動的コンパイル方式によるバイトコード実行の高速化の効果は少ないと考えられる。

バイトコードの実行ステップ数が異なる二つの測定プログラムを用意し、実行速度の向上率(表 1)を測定した。

1）アプリケーション

計測項目「配列代入とソート」では、11.31 倍の性能向上が得られた。測定時間のほとんどは、アプリケーションに含まれるバイトコードの実行時間である。そのため、Java バイトコードをネイティブコードに変換する動的コンパイル方式を採用することで大きな効果が得られる。

2）描画クラスライブラリ

測定プログラムは、描画クラスライブラリの矩形描画を行うメソッドの処理時間を測定している。矩形描画メソッドは、携帯端末のシステムに含まれる描画ライブラリの矩形描画関数を呼び出すために、測定時間内のバイトコードの実行時間が占める割合は少ない。そのため、性能向上は 1.31 倍であった。

表 1 動的コンパイル方式の高速化

測定分類	計測項目	向上率
アプリケーション	配列代入とソート	11.31 倍
描画クラスライブラリ	矩形描画	1.31 倍

図 1 ゲーム操作画面

3. Java アプリケーションの特性と関連する実装

高速化手法を検討する上で着目した Java アプリケーションの特性とそれに関連する Java クラスライブラリの実装を述べる。

3.1 描画処理の特性

Java アプリケーションは、多数の小さな画像や文字列を描画することで操作画面を構成している。例えば、図 1 に示したゲーム操作画面では、敵キャラクタを 3 回、戦闘機を 1 回、敵キャラクタのミサイルを 2 回、バックグラウンドの星を 4 回の計 10 回の画像描画が行われる。

Java アプリケーションは、Java クラスのフィールドに、位置、回転や反転等の処理内容を設定し、メソッドを呼び出すことで画像/文字列を描画する。呼び出された Java クラスの描画処理メソッドは、Java アプリケーションが設定したフィールドの格納データを参照し、要求された処理を行う。そのため、Java クラスのフィールドの参照方法を高速化することで、描画処理の高速化を実現することができる。

3.2 読み込み/書き込み処理の特性

Java アプリケーションは、Java 実行環境が提供するデータ領域に読み込みと書き込みを行う。例えば、ゲームでは、画像やゲーム進行状況、ゲームの点数を書き込み、表示する際に読み込みを行う。

Java クラスライブラリは、データ領域の読み込み/書き込み速度を向上するために、データ領域のオープン時に低速なファイルシステムから高速なメモリにデータを複製し、クローズ時に複製データを破棄している。しかし、Java アプリケーションがオープン・クローズ状態を管理せずに、読み込み/書き込み毎にオープン・クローズ処理を行っている場合には、オープン時に行うデータの複製処理が頻繁に発生し、性能に大きな影響を与える。

Acceleration of Java Application for Hand-held Device
Katsuhide TAKAHASHI
Mitsubishi Electric Corporation Information Technology R&D Center

携帯端末に格納できるJavaアプリケーションのサイズには制限が設けられていることが一般的であり、Javaアプリケーションのコード量は少なくしなければならない。そのため、Javaアプリケーションは、コードを削減するためにオープン・クローズ状態を管理せずに、読み込み/書き込み処理を行うと考えられる。

4. 描画処理の高速化

Java クラスの描画処理メソッドは、処理内容が設定されたフィールドのデータを参照するために、名前(クラス名、フィールド名)をキーとして、格納したアドレスをJavaVM内のデータベースから検索する処理を行っている。今回、図 2 に示すように、頻繁にアクセスするフィールドの格納アドレスを特定するための情報を保持することで、Javaクラスのフィールドデータの参照を高速化し、描画処理の高速化を実現する。

5. 読み込み/書き込み処理の高速化

Java アプリケーションは読み込み/書き込み毎にオープンとクローズを行うため、データ領域の読み込み/書き込み速度の向上を目的としたオープン時のデータ複製処理は性能を低下させる。そのため、Javaクラスライブラリがクローズ時に行っている複製データの破棄を見直すことで、読み込み/書き込み処理の高速化を実現することができる。

クローズした後も複製データを保持することで、オープン時のデータ複製の処理時間を削減することが可能であるが、複製データを保持するのみの実装では、メモリ使用量を増加させる。そのため、図 3 に示すように、メモリ不足が発生した場合に、クローズ後に保持している複製データを破棄するように実装することで、メモリ使用量を増加させずに読み込み/書き込み処理の高速化を実現する。

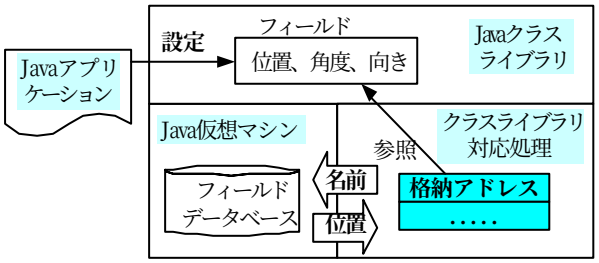

図 2 描画処理の高速化

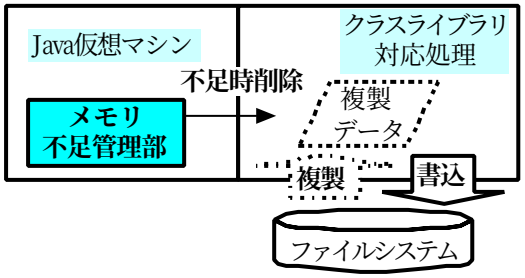

図 3 読み込み/書き込み処理の高速化

表 2 描画処理の高速化

計測項目	画像サイズと描画回数	向上率
画像描画（小）	16x16ピクセル: 225回 32x32ピクセル: 20回	1.64 倍
画像描画（大）	240x240ピクセル: 1回 80x80 ピクセル: 9回	1.15 倍

表 3 描画処理の高速化

計測項目	処理内容	向上率
読み込み	10Kバイトを20回	24.36 倍
書き込み	10Kバイトを20回	1.92 倍

6. 評価

6.1 描画処理

描画処理の高速化手法を評価するために、大きさの異なる画像を描画する測定プログラムを用意し、適用前後の実行速度の向上率(表 2)を測定した。

計測項目「画像描画(小)」では、1.64 倍の性能向上が得られた。描画処理の高速化手法は、描画に伴うデータ参照の処理時間を短縮するものである。そのため、計測項目「画像描画(大)」では、画像データの画面への転送時間を含めた描画処理の全体に対するデータ参照の処理時間の占める割合は小さく、性能向上は1.15 倍程度となった。

携帯端末のゲームに用いる画像サイズは小さく、計測項目「画像描画(小)」の画像程度である。そのため、描画処理の高速化手法は有効であり、1.6 倍の高速化を実現できる。

6.2 読み込み/書き込み処理

読み込み/書き込み処理の高速化手法を評価するために、読み込み/書き込み毎にデータ領域のオープン・クローズを行う測定プログラムを用意し、適用前後の実行速度の向上率(表 3)を測定した。

計測項目「読み込み」では、24.36 倍の大きな性能向上が得られた。計測項目「書き込み」では、書き込まれたデータをファイルに反映させる処理が含まれるため、削減したデータ複製処理の時間が占めていた割合は小さく、1.92 倍の性能向上となった。

7. おわりに

携帯端末上の Java 実行環境の Java クラスライブラリに、Javaアプリケーションの特性を利用した高速化手法を適用することで、実行速度の向上が得られることを確認した。

今後は、検討した以外の処理、例えば、3D 描画等について、携帯端末上のJavaアプリケーションの特性を利用した高速化手法の検討を行う予定である。

参考文献

[1] Lindholm, The Java Virtual Machine Specification, 2nd Edition, Sun Microsystems, Inc.
[2]高橋他,携帯端末向けJavaの高速化手法の検討,情報処理学会研究報告,Vol. 2002,MBL-22-11, Oct.2002
[3]高橋他,携帯端末Javaの高速化手法の検討と評価,情報科学技術フォーラム2003,M-020,Sep.2003