Androidアプリケーションの起動性能の解析システム

永田恭輔† 山口実靖†
†工学院大学 工学部 情報通信工学科

本論文では，Androidのアプリケーション起動時間の解析システムを提案する．提案システムを実装し，起動時間の解析を行った結果，提案システムによりアプリケーションの起動時間の詳細な調査が可能であることが確認された．

An Android Application Analyzing System

Kyosuke Nagata† Saneyasu Yamaguchi†
†Kogakuin University

In this paper, we propose a method for analyzing android application starting performance. Our implementation demonstrated that the system was useful to discuss performance.

1. はじめに

近年，携帯端末向けプラットフォームであるAndroid の普及が進んでおり，携帯電話をはじめとする様々なハードウェアで利用される様になってきている．

Android OS を搭載した携帯電話端末(スマートフォン)は従来の携帯電話端末より多くの機能を備えているが，アプリケーション性能は必ずしも十分ではなく，性能に関する考察が行われている[1]．携帯電話端末ユーザにとって重要な性能の一つにアプリケーションの起動時間があげられる．しかし，起動処理にはAndroid OSによる処理とアプリケーションによる処理の両方が含まれており，既存のベンチマークやプロファイラーはこの性能を考察するのに適さない．また，Android システムは発展途上の段階にあり，起動時間の長さがOSやアプリケーションのどの部位に起因するものであるかは必ずしも明確には考察されてない．そこで本論文では，OSとアプリケーションを統合的に解析しアプリケーションの起動時間を詳細に解析できるシステムを提案し，その動作を示す．

2. Androidアプリケーション

Android OS では，Dalvik VM で実行されるアプリケーションは以下の手順により起動される．まず，アプリケーションの起動が要求される．次に，新しいプロセスがZygoteからforkされて生成される．そして，Android アプリケーションのライフサイクルにて定められる onCreate()，onStart()，onResume()の各メソッドが実行され，アプリケーションは起動済みと言える状態となる．

Zygoteは OS 起動時に生成される最初の Dalvik VMのプロセスである．全てのAndroidアプリケーションはZygoteプロセスをforkすることにより生成される．Zygoteがいくつかのクラスをすでにロードしているため，Zygote を fork して生成されたプロセスは生成時からそれらのクラスをロード済みの状態となる．これにより，アプリケーションの起動時間の短縮が実現されている．

3. 解析システム

本章では，アプリケーション起動に含まれる各処理に要する時間を調査し，起動性能の考察を可能とするシステムを提案する．

Android システムがオープンソースであることを利用し，Android カーネルとアプリケーションフレームワークに以下の機能を追加する．まずAndroidカーネルに，fork()関数が呼び出されプロセスが生成された際に時刻とプロセス ID をメモリに保存する機能を追加する．また，アクティ

ビティマネージャサービスの startProcessLocked()メソッド，Dalvik VM の dvmInitAfterZygote()関数，app_main の onZygoteInit()関数， Java フレームワークのライフサイクルメソッド群(onCreate()など)が呼び出された時刻を Android ログシステムを用いて記録する機能を追加する．

上記機能が追加された Android 上にて，時刻取得プログラム(native 実装)を実行しアプリケーション起動開始直前の時刻を取得する．そして，その直後に Android 標準コマンド am を用いてアプリケーションの起動要求インテントを送出する．これにより，アプリケーション起動に含まれる各処理に要した時間を知ることができる．

4. 解析結果

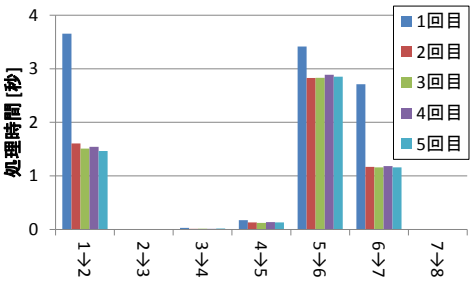
提案システムを実装し， HT03A， Android 2.1-update1 を用いてアプリケーション起動時間の解析を行った．ブラウザアプリケーションを 5 回連続で起動しその起動時間を解析した結果を図 1 に示す．ブラウザアプリケーション 4 回起動と，ゲームアプリケーション 4 回起動を交互に行ったときの解析結果を図 2 に示す．両解析とも，アプリケーションを起動して測定が終わるたびにプロセスを終了(KILL)してから次の起動を行っている．

両解析結果より，アプリケーション起動の内の最初の処理，5 番目の処理，6 番目の処理に多くの時間を要していることを確認することができる．また，2 回目以降の起動では各処理の時間が短縮されていることなどを知ることができる．

また，同一アプリケーションの起動時間は 2 回目以降短縮されるが，図 2 よりその効果が他のアプリケーションの起動を挟んでも継続されることが多いことを確認できる．

5. 考察

提案手法の測定精度と負荷について考察する．fork()呼び出しの調査はカーネル内にて行われ，精度も高く，負荷(時刻取得とメモリコピー)も小さいと期待される．dvmInitAfterZygote()と



1=Intent 時刻， 2=fork 時刻， 3=StartProc 時刻，4=dvmInitAfterZygote 時刻， 5=onZygoteInit 時刻，6=onCreate 時刻， 7=onStart 時刻

図 1 ブラウザ起動の解析

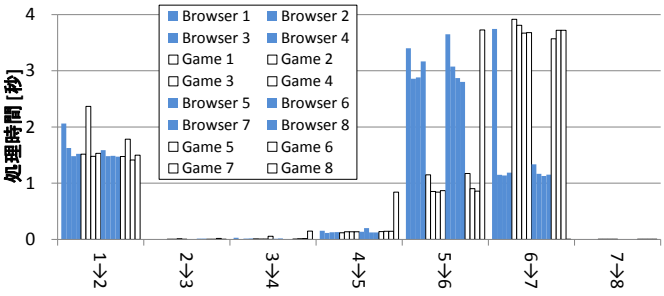


図 2 ブラウザとゲームの起動の解析結果

onZygoteInit()関数の調査は，native 実装内で時刻取得が行われ精度は高いが，ログ機能を用いており負荷は改善の余地があると考える．startProcessLocked()メソッドとライフサイクルメソッドの調査は Dalvik VM 上で行われ，上記より精度が劣り，負荷が大きいと予想される．

6. おわりに

本論文では，Android アプリケーションの起動時間の調査方法を提案し，その実装を用いた解析例を示した．今後は，本解析システムを用いてアプリケーションの性能改善を実現していく予定である．

謝辞

本研究は科研費 (22700039) の助成を受けたものである．

参考文献

[1] 間嶋 崇，横山 哲郎，曾 剛，神山 剛，冨山 宏之，高田 広章，"Android プラットフォームにおける Dalvik バイトコードの CPU 負荷量の解析"，情報処理学会研究報告．UBI，vol. 2010，No. 18，pp. 1-8 (2010)