AndrodiOS におけるアプリケーション起動時間の解析

西中 一志[†1]　森 竜佑[†1]　村上 翼[†1]　神山 剛[†2]
福田 晃[†2]　小口 正人[†3]　山口 実靖[†1]

キーワード:Android，アプリケーション起動時間，アプリケーションライフサイクル

1. はじめに

近年，スマートフォンが普及し重要なプラットフォームとなっている．特に Android スマートフォンは世界的に高いシェアで普及しており，特に重要なプラットフォームとなっている．よって，同プラットフォームの性能向上は，重要な課題であると言える．アプリケーションの起動時間はユーザーエクスペリエンス向上のための重要な要素の一つである．しかし，一部のアプリケーションでは起動時間が非常に長いという問題がある．

本稿では，アプリケーションライフサイクルに着目し，アプリケーションライフサイクルの中でもアプリケーションの起動に関わる処理である onCreate()，onStart()，onResume()での各処理時間の解析手法を提案する．

2. 関連研究

関連研究として Android アプリケーション起動時間の評価[1)2)]がある．この研究では，onCreate()などの一部の処理の時間の取得は達成されているが，onResume()の処理の時間などは取得できておらず，網羅的にアプリケーション起動時間を観察することはできない．

本稿で提案する手法は，onResume()の処理の開始時刻，終了時刻(そしてこれらから計算される処理時間)も含めて取得するため，ライフサイクルにおけるアプリケーション開始処理の時間を網羅的に解析することができる．

文献3)4)にて，Android OS のカーネル内部の処理をモニタリングする手法が提案されているが，アプリケーション起動時のライフサイクルの観察は実現されていない．ただし，当該手法を用いて本稿の提案手法を拡張することにより，さらに詳細な観察が可能になると期待できる．

†1 工学院大学
Kogakuin University
†2 九州大学
Kyusyu University
†3 お茶の水女子大
Ochanomizu University

3. Android アプリケーションライフサイクル

Android においてアプリケーションの起動要求を発行すると，ActivityManager へ起動要求の Intent が送出され，Zygote プロセスがアプリケーションプロセスをフォークし，アプリケーションプロセスが初期化され，アプリケーションライフサイクルの onCreate()，onStart()，onResume()が実行される．

アプリケーションライフサイクルは図 1 アプリケーションライフサイクルの通りである．図のようにアプリケーションライフサイクルにはアクティビティの開始に関わるメソッドである onCreate()，onStart()，onResume()，onRestart()と，アクティビティの終了に関わるメソッドである onPause()，onStop()，onDestroy()が存在する．

4. 提案手法

本章にて，Android アプリケーションの起動時に実行されるライフサイクル処理にかかる時間の解析手法を提案する．

アプリケーションの onCreate()や onStart()のメソッドがよばれると，Android アプリケーションフレームワーク Activity.java の onCreate()や onStart()のメソッドが呼び出される．そこで，提案手法では Activity.java 内のこれらメソッドが開始された時刻を取得し，ログに記録するようにする．開始された時刻の取得は，Activity.java の

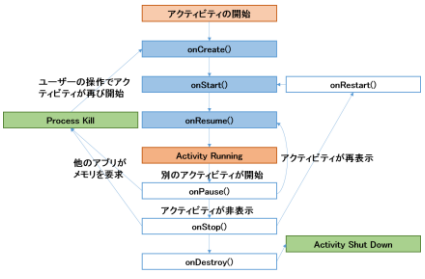

図 1 アプリケーションライフサイクル

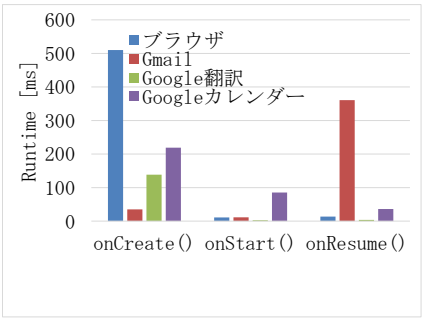

図 2　起動時間解析結果

onCreate()や onStart()の最初の行に時刻取得メソッド(System. currentTimeMillis())を記述することにより実現できる.

アプリケーションの onResume()メソッドは, Instrumentation.java 内 callActivityOnResume()メソッド内で呼ばれるため, 同メソッド内におけるアプリケーションの onResume()メソッド呼び出しの直前と直後の時刻を取得するようにする.

これらの時刻の取得により, アプリケーションの onCreate(), onStart(), onResume()に要した時間が計測可能となる. Activity.java の onCreate()の開始時刻から onStart()の開始時刻までが, アプリケーションの onCreate()メソッドに要した時間である. 同様に, Activity.java の onStart()開始から callActivityOnResume()内のアプリケーション onResume()呼び出しの直前までの時間が, アプリケーション onStart()に要した時間である. そして, アプリケーション onResume()呼び出しの直前と直後の時間がアプリケーションの onReume()処理に要した時間である.

5. 解析結果

前章の提案解析手法を用いて, 実アプリケーションの起動時間の解析を行い, 提案手法の有効性を検証した. 起動時間の計測を行ったアプリケーションは, Android 標準ブラウザ, Gmail, Google 翻訳, Google カレンダーの 4 つである. それぞれのアプリケーションで 3 回ずつ測定をした. 図 2　起動時間解析結果の計測結果を示す. 縦軸は, 各アプリケーションの各処理に要した時間の平均である.

図 2　起動時間解析結果から, Gmail を除くアプリケーションにおいて, onCreate()が最も時間を要しており onStart()や onResume()には殆ど時間を要さないことが分かる. この傾向が最も顕著に表れているのはブラウザであり onCreate()での時間も他のアプリケーションと比べ著しく長くなっている. Gmail はこの傾向がなく, onCreate()と onStart()が短く, onResume()にかかる時間が非常に長くなっている. これらの解析結果から, アプリケーション開始の各ライフサイクルに要する時間は異なることが分かる. また, 提案手法により, アプリケーションごとの開始処理の特徴を観察できることが分かった.

6. おわりに

本稿ではアプリケーションライフサイクルに着目し, Android アプリケーションの起動時間の解析システムを提案した. そして, 実アプリケーションの起動時間の解析結果を示し, 提案システムの有効性を示した. 実験の結果, アプリケーションによって起動時間の傾向に違いがあることが分かり, 提案手法の有効性が確認された.

今後は, より多くのアプリケーションでの実験や, カーネル内部の動作の観察の実現を行う予定である.

謝辞

本研究は JSPS 科研費 15H02696, 17K00109, 18K11277 の助成を受けたものである.

本研究は、JST、CREST JPMJCR1503 の支援を受けたものである.

参考文献

1) Kyosuke Nagata, Saneyasu Yamaguchi, "An Android application launch analyzing system," 2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT), Seoul, 2012, pp. 76-81.

2) K. Nagata, Y. Nakamura, S. Nomura and S. Yamaguchi, "Measuring and Improving Application Launching Performance on Android Devices," *2013 First International Symposium on Computing and Networking*, Matsuyama, 2013, pp. 636-638. doi: 10.1109/CANDAR.2013.116

3) Yuta Nakamura, Kyosuke Nagata, Shun Nomura, and Saneyasu Yamaguchi. 2014. I/O scheduling in Android devices with flash storage. In *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication* (ICUIMC '14). ACM, New York, NY, USA, , Article 83 , 7 pages. DOI: https://doi.org/10.1145/2557977.2558025

4) H. Hirai, M. Oguchi and S. Yamaguchi, "A proposal on cooperative transmission control middleware on a smartphone in a WLAN environment," *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Lyon, 2013, pp. 693-700. doi: 10.1109/WiMOB.2013.6673432