B-019

時間加速 Android 環境におけるシステム安定性に関する一考察
A Study on System Stability on An Accelerated Android

福田 翔貴† 栗原 駿† 小口 正人‡ 山口 実靖†
Shoki Fukuda Shun Kurihara Masato Oguchi Saneyasu Yamaguchi

1. はじめに

スマートフォンやタブレット PC などの普及により，膨大な数の Android OS 向けアプリケーションが開発，配布されており，アプリケーションの動作解析の重要性が増している．しかし，アプリケーションを実際に動作させて振る舞いを観察する動的動作解析には長い時間がかかる問題がある．この問題に対して，我々は過去の研究にてカーネルの時刻管理実装を改変しシステム内の時間の流れを加速することにより観察時間を短縮する手法を提案[1]し，低加速倍率環境では安定して動作することを示した[2]．本稿では高倍率加速環境を構築しその安定性の評価を行い，高加速倍率環境委における課題とその解決策についての考察を行う．

2. 端末内時間の加速

Android OS における時間の管理は Linux カーネルにより行われる．したがって Linux カーネルを改変することにより，Android 端末内部のシステムが認識する時間の流れを速くすることができる．

Linux カーネルには cycle_now という変数があり，ここにはクロックソースから取得された値が格納されている．この値は時間の経過とともに増加を続けており，この増分を n 倍にすることで，端末内部のシステムが認識する時間が n 倍速で進む時間加速環境を実装可能である．

加速環境の動作検証は既存研究[2][3]にて行われているが，いずれも低倍率加速にて評価が行われており，より高倍率の加速環境による評価が必要であると考えられる．

3. 評価

本章にて Android OS における高倍率加速環境の評価を行う，

3.1 高倍率加速環境における観察精度評価

加速率と観察精度の関係について評価する．提案手法を適用した Android 端末内時計加速環境の加速倍率を 2, 6, 24, 48, 60 倍と変化させ，加速倍率が測定結果に与える影響を調査した．測定は，実験用アプリケーションをインストールし無操作状態で端末を 1 時間放置し，その間に実行される Alarm セットと WakeLock を観察することにより行った．実験用アプリケーションには，2015 年 10 月 24 日 GooglePlay ニュース&雑誌カテゴリ上位 10 件を使用した．計測環境は表 1 のとおりである．Alarm セットと WakeLock は，その観察がバッテリ消費を増やすアプリケーションの特定に重要であることが確認されている[3]ため，これらに着目してアプリケーションの観察を行った．

観察結果を図 1 から図 3 に示す．図 1 より 6 倍以下の加速では非加速状態（1 倍）と加速状態の観察結果がほぼ等しく，加速による観察時間の短縮が高い精度で実現されていることがわかる．また 24 倍以上の加速倍率においては，Alarm セット回数の通常状態と加速状態の測定結果の差異が大きくなり，WakeLock の観察回数は加速倍率の上昇に伴い単調に減少していることがわかる．図 2 および図 3 からも同様に，加速倍率が上昇するごとに Alarm セット間隔や WakeLock 実行間隔の計測結果の加速有無の差異が拡大していることが分かる．なお，確認された最大の差異は Alarm セット回数 14.9%，WakeLock 回数 26.6%（いずれも加速率 60 倍時）であり，要求される精度がこれより低い状況では 60 倍の加速による観察も可能であることが分かる．

また，既存研究[3]の目的である，頻繁に Alarm セットや WakeLock を行うアプリケーションの検出は達成しており，同研究で求められる精度は 60 倍の加速でも得られていると言える．

表 1 計測環境

Device Name	Nexus7 (2013)
OS	Android 5.0.1 (aosp) (kernel 改変済)

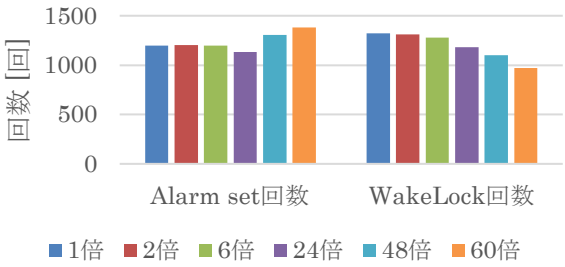

図 1 加速倍率と Alarm セット，WakeLock 回数

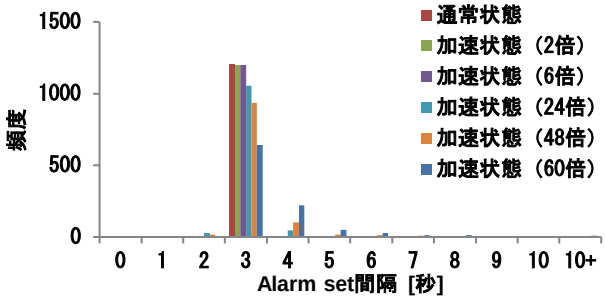

図 2 加速倍率と Alarm セット間隔

† 工学院大学大学院 工学研究科 電気・電子工学専攻
Electrical Engineering and Electronics, Kogakuin University Graduate School
‡ お茶の水女子大学 理学部 情報科学科
Department of Information Sciences, Ochanomizu University

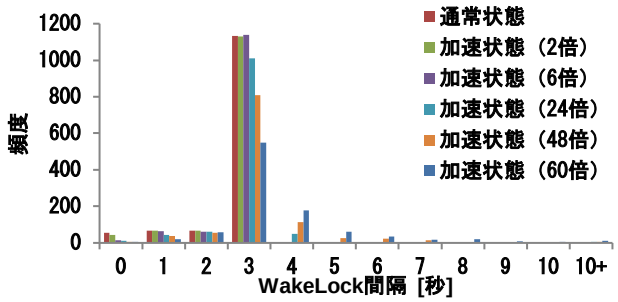

図 3 加速倍率と WakeLock 間隔

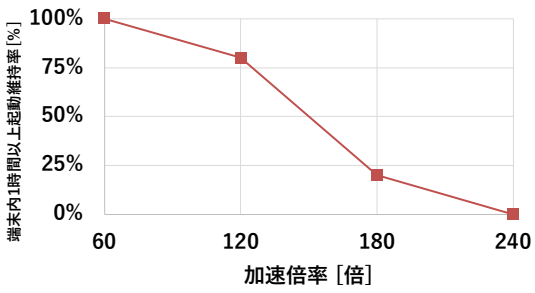

図 4 加速倍率とシステム安定性（対策なし）

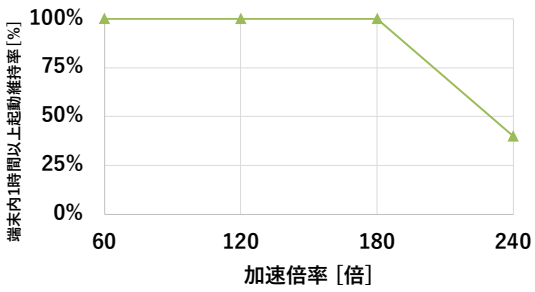

図 5 加速倍率とシステム安定性（対策あり）

3.2 高倍率加速環境における安定性評価

加速率とシステムの安定性の関係について述べる．各加速率において Android 端末を無操作状態のホーム画面表示にて放置し，安定して動作した時間を計測した．計測ごとに OS の再起動を実行し，OS 起動後 30 秒経ってから時間加速を有効化した．測定環境は表 2 のとおりである．

実験結果を図 4 に示す．図の縦軸は端末内時間で 60 分以上障害が起きなかった割合を示している．図 4 より，おおよそ 60 倍までの加速率においてはランチャー画面表示を維持できることが分かる．また，加速倍率の上昇に従いシステムの安定性が低下することも確認された．

発生した障害の程度であるが,すべての加速率にて Linux カーネルの停止は確認されず，ADB により接続した shell はすべての倍率の測定にて動作を続けることができた．図内の障害はすべて Android OS のユーザ空間部プロセスの再起動となっている．これは，Zygote などの Android アプリケーションフレームワークプロセスが停止し,フレームワークが再起動している状況である．

次に，システムの障害を招いた原因についての考察を行う．Android のユーザ空間アプリケーションフレームワーク部が再起動した状況における Android OS のログには，Slow operation の検出やウォッチドッグタイマなどの各種タイマのタイムアウトの検出が発生し,プロセスが強制的に再起動されていることが記録されている．これは，CPU などの処理能力が加速されないまま時間の流れのみが加速され，各種処理に要した時間が非加速状態より長く認識され，通常であれば検出されない Slow operation やタイムアウトが検出されてしまったことが原因であると考えられる．

参考のために，Android OS を改変しウォッチドックタイマや ActivityManager によるシステムプロセスの強制終了を無効化する対策を行い，観察を行った．

観察結果を図 5 に示す．図より，対策により起動状態を維持できる確率が高まっていることが分かる．

表 2 計測環境

Device Name	Nexus7 (2013)
OS	Android 5.1.1 (aosp) （kernel 改変済）

4. おわりに

本稿では，Android アプリケーション動的解析にかかる時間を短縮することの重要性と，端末内部の時間の流れを加速する手法を紹介した．そして，既存研究においては低倍率加速環境における評価のみが行われていることを考慮し，本稿においては高倍率における評価を行った．評価の結果，高倍率加速環境においてはある程度の誤差が生じること，既存研究で求められている測定精度は得られること，倍率の増加に伴いシステムの安定性が低下することが確認された．そして，システムの安定性はウォッチドッグタイマや ActivityManager によるシステムプロセスの強制終了を無効化する対策を行うことにより改善することが確認できた．

今後はより高い倍率での評価やシステムの安定性の向上に取り組んでいく予定である．

謝辞

本研究は JSPS 科研費 26730040, 15H02696, 17K00109 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] Shoki Fukuda, Shun Kurihara, Shintaro Hamanaka, Masato Oguchi, Saneyasu Yamaguchi, "An Accelerated Application Monitoring Environment with Accelerated Servers," The 5th IEEE Global Conference on Consumer Electronics (GCCE 2016), 2016.

[2] Shoki Fukuda, Shun Kurihara, Shintaro Hamanaka, Masato Oguchi, Saneyasu Yamaguchi, "Accelerated test for applications with client application and server software," ACM IMCOM 2017: International Conference on Ubiquitous Information Management and Communication, 100, 2017.

[3] Shun Kurihara, Shoki Fukuda, Shintaro Hamanaka, Masato Oguchi, Saneyasu Yamaguchi, "Application power consumption estimation considering software dependency in Android," ACM IMCOM 2017: International Conference on Ubiquitous Information Management and Communication, 86, 2017.